FP7-610416-PANOPTESEC

Dynamic Risk Approaches for Automated Cyber Defence

D4.3.2: Data Collection and Correlation Integration Prototype Report

Work-Package	WP4	Deliverable	D4.3.2
Due Date	2016-06-30	Submission Date	2016-06-30
Main Author(s)	UzL		
Version	V1.0	Status	final
Dissemination level	PU	Nature	R
Keywords	Integration Report, Data Collection and Correlation, Scientific Contribution, Experimentation and Integration Results		

Part of the Seventh
Framework Programme

Funded by the EC - DG Connect

EXECUTIVE SUMMARY

With the main components discussed in this deliverable (NIP, VIP, PM, MIM, RMC, and LLC) we were able to show that one can use a generic architecture (as part of the PANOPTESEC system) to successfully collect data for high-level online correlation (instantiation of attack paths, WP5) as well as for analyzing suspicious activities in general. This was achieved by providing an architecture that regularly updates context information (from ACEA) in order to drive online data collection and correlation (LLC). Scalability of the integration data collection correlation system was shown using experiments with the so-called emulation system based on the ACEA contingency planning and emergency handling cyber-physical system. Based on results of WP4, further correlation systems (see WP5 of PANOPTESEC) as well as the visualization system(s) (WP6) were able successfully carry out their jobs.

Presented results include results obtained from individual components, such as local performance, scalability and functional tests, as well as results obtained from integrated components, such as results that require an interaction between WP4 and WP5 or WP6 components. Moreover, we discuss, present and validate results obtained from individual and integrated components from the ACEA use case and discuss relations between production and emulation data for some components. Finally, we present scientific results that have been published as novel scientific contributions.

Verification, validation and integration tests of the WP4 components delivered highly satisfying results for the ACEA use case partner and provided valuable, well-accepted scientific contributions accepted at major conferences, such as IJCAI 2015, ARES 2016, and AI 2015.

HISTORY

Version	Date	Name/Partner	Comments
V0.1	13th May	Alexander Motzek/UZL	Creation of Document
V0.2	26th May	Dennis Heinrich/UZL	Restructuring of Document
V0.3	5th June	Alexander Motzek/UZL	Draft Introduction
V0.4	7th June	Stefan Werner/UZL	Extended Appendix
V0.5	16th June	Alexander Motzek/UZL	Contribution to MIM, NDA and Integration
V0.6	17th June	Matteo Merialdo/RHEA	Contribution to Data Collection
V0.7	17th June	Mona Lange/UZL	Contribution to LLC Chapter
V0.7.1	19th June	Dennis Heinrich/UZL	Updated Appendix Minor Changes
V0.8	19th June	Luca Severini/EPIST	Contribution to VIP, NIP and RMC
V0.9	20th June	Dennis Heinrich/UZL	Document for QA
V1.0	29th June	Ralf Möller/UZL	Final Version after QA
V1.1	31th August	Ralf Möller/UZL	Update to be conform to D4.2.2

TABLE OF CONTENTS

1 Introduction **10**

1.1 Context . . . 10
1.2 Purpose . . . 10
1.3 Scope . . . 10
1.4 Document Structure . . . 10

2 Methodology **12**

2.1 Selection of Test Data . . . 12
2.2 Dissemination Activities . . . 12
2.3 Quality Assurance . . . 13

3 Data Collection **14**

3.1 Data Collection Related Data Model . . . 15
3.1.1 Network Inventory . . . 15
3.1.2 Vulnerability Inventory . . . 15
3.1.3 Scored Vulnerability Inventory . . . 18
3.1.4 Reachability Matrix . . . 20
3.2 Network Inventory Processor . . . 24
3.3 Vulnerability Inventory Processor . . . 27
3.3.1 Vulnerability Information Data Sources . . . 27
3.3.2 Patch Management Information . . . 29
3.3.3 Port and Protocol Information . . . 30
3.3.4 VIP Implementation for Patches and Ports/Protocols . . . 30
3.4 Reachability Matrix Correlator . . . 31
3.5 Integration of Components and Evaluation of Performance . . . 32

4 Low-Level Correlation **45**

4.1 Introduction . . . 45
4.2 Event Processing . . . 46
4.2.1 Event Normalization . . . 46

4.2.2 Event Verification . . . 47
4.3 Evaluation . . . 52
4.3.1 Network Service Dependencies within the Emulation Environment . . . 58
4.3.2 Performance Evaluation of Alert Processing Time . . . 61
4.3.3 Performance Evaluation of Network Dependency Analysis . . . 63
4.4 Discussion . . . 64

5 Mission Impact Modeling and Mission Impact Assessment 66

6 Conclusions 71
6.1 Significant results achieved . . . 71
6.2 Recommendations . . . 71
6.3 Deliverable validation . . . 71

Appendices 72

A A Semantic Approach to Reachability Matrix Computation 73
A.1 Introduction . . . 73
A.2 The Knowledge Base . . . 75
A.3 About the Reasoning . . . 76

B Event Prioritization and Correlation based on Pattern Mining Techniques 79
B.1 Introduction . . . 79
B.2 Related Work . . . 80
B.3 Network Model . . . 81
B.4 Network Dependency Analysis . . . 83
B.4.1 Direct Dependency . . . 83
B.4.2 Indirect Dependency . . . 84
B.4.3 Clustering Dependencies . . . 85
B.4.4 Tasks . . . 86
B.5 Event Processing . . . 86
B.5.1 Event Normalization . . . 86
B.5.2 Event Prioritizing and Correlation . . . 87
B.6 Experimental Evaluation . . . 87
B.7 Conclusion . . . 88

C Time Series Data Mining for Network Service Dependency Analysis 91
C.1 Introduction . . . 91
C.2 Related Work . . . 92
C.3 IT Network Model . . . 93
C.4 Network Service Dependency Discovery . . . 95
C.4.1 Communication Histograms . . . 95
C.4.2 Indirect Dependencies . . . 97
C.4.3 Measuring Communication Histogram Similarity . . . 98
C.5 Experimental Evaluation . . . 99
C.5.1 Comparable Network Dependency Analyzers . . . 100
C.5.2 Analysis Methods . . . 101
C.5.3 Performance and Sensitivity Evaluation . . . 101
C.6 Conclusion . . . 103

D Using a Deep Understanding of Network Activities for Network Dependency Analysis 107
D.1 Introduction . . . 107
D.2 Related Work . . . 108
D.3 Network Service Dependency Discovery . . . 109
D.3.1 Network Model . . . 109
D.3.2 Direct Dependencies . . . 110
D.3.3 Indirect Dependencies . . . 110
D.4 Workflow Mining . . . 112
D.5 Experimental Evaluation . . . 114
D.5.1 Network Simulation . . . 114
D.5.2 Method for Analyzing the Performance of Network Service Dependency Discovery . . . 115
D.5.3 Performance Evaluation . . . 115
D.6 Conclusion . . . 118

E Context- and Bias-Free Probabilistic Mission Impact Assessment 122
E.1 Introduction . . . 122
E.2 Dependencies and Impacts . . . 123
E.2.1 Mission Dependency Model (Business View) . . . 124
E.2.2 Resource Dependency Model (Operation View) . . . 126

E.2.3 Local Impacts (Security View) . . . 128
E.3 Probabilistic Mission Impact Assessment . . . 131
E.3.1 Monte-Carlo Approximation . . . 132
E.4 Complexity and Experimental Evaluation . . . 134
E.4.1 Complexity Analysis and Experiments . . . 134
E.4.2 Usecase Experiments . . . 137
E.5 Extensions to Dynamic Mission Impact Assessment . . . 140
E.6 Related Work . . . 143
E.7 Conclusion . . . 144

F Selection of Mitigation Actions Based on Financial and Operational Impact Assessments 148
F.1 Introduction . . . 148
F.2 Financial Impact Assessment . . . 149
F.3 Operational Impact Assessment . . . 150
F.3.1 Mission Dependency Model (Business View) . . . 151
F.3.2 Resource Dependency Model (Operation View) . . . 152
F.3.3 Local Impacts (Security View) . . . 152
F.3.4 Mathematical Mission Impact Assessment . . . 153
F.4 Dynamic Risk Management Response System . . . 154
F.4.1 Response Financial Impact Assessor (RFIA) . . . 154
F.4.2 Response Operational Impact Assessor (ROIA) . . . 157
F.4.3 Selection of Response Plans . . . 158
F.5 Use Case . . . 159
F.5.1 Threat Scenario . . . 161
F.5.2 Financial Impact Assessment . . . 161
F.5.3 Operational Impact Assessment . . . 162
F.5.4 Selection of Mitigation Actions . . . 164
F.6 Related Work . . . 164
F.7 Conclusion . . . 165

G Semantic Normalization and Merging of Business Dependency Models **168**

G.1 Introduction . . . 168
G.2 Business Dependency Models . . . 169
G.3 Merging and Matching Problems . . . 171
G.4 Experiments and Discussion . . . 176
G.5 Conclusion . . . 178

H Indirect Causes in Dynamic Bayesian Networks Revisited **180**

H.1 Introduction . . . 180
H.2 Dynamic Bayesian Networks: Preliminaries . . . 181
H.3 Activator Dynamic Bayesian Networks . . . 183
H.4 Operations . . . 188
H.4.1 Filtering . . . 188
H.4.2 Smoothing . . . 189
H.5 Discussion . . . 190
H.6 Conclusion . . . 191

Table 1: Acronym List

Acronym	Meaning
ABE	Automata Based Engine
ACEA	ACEA S.p.A
BMC	Business Mission Collector
BN	Bayesian Network
BPMN	Business Process Modeling Notation
CVE	Common Vulnerabilities and Exposures
DBN	Dynamic Bayesian Networks
DCC	Data Collection Collector
DCI	Data Collection Interface
DPGM	Dynamic Probabilistic Graphical Model
DMIA	Dynamic Mission Impact Assessment
HLD	High Level Design
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
LLC	Low Level Correlator
MG	Mission Graph
MIA	Mission Impact Assessment
MIM	Mission Impact Module
MONA	Mission Oriented Network Analysis
NI	Network Inventory
NIP	Network Inventory Processor
NIPA	Network Inventory Processor Agent
PGM	Probabilistic Graphical Model
PM	Persistency Manager
QBE	Query Based Engine
RFIA	Response Financial Impact Assessment
RMC	Reachability Matrix Correlator
RORI	Return On Response Investments
SCADA	Supervisory Control and Data Acquisition
SIEM	Security Information and Event Management
SNDA	Shallow Network Dependency Analyzer
SNDM	Shallow Network Dependency Model
SRD	Strategic Response Decider
UzL	University of Lübeck
VIP	Vulnerability Inventory Processor
WP	Work Package

1 INTRODUCTION

1.1 Context

The data collection and correlation (DCC) component developed in WP4 is composed of multiple modules, namely, the NIP, VIP, PM, MIM, LLC, and RMC. All modules are based on state-of-the-art and beyond-state-of-the-art approaches. In this document, we highlight, discuss and relate the scientific contributions of beyond-state-of-the-art approaches, which are documented in scientific articles and papers.

In the following sections we highlight main scientific contributions of the modules related to the PANOPTESEC project. All referenced documents are given as appendices. All articles provide valuable results obtained from the integrated WP4 component of the PANOPTESEC project and contain results from individual modules of WP4. We introduce all mathematical foundations required in each module and present results that show the applicability of WP4 components and their integrated behaviour to the ACEA use case and, further, to different application fields.

1.2 Purpose

The purpose of this deliverable is to present scientific outcomes and technical details from a general point of view of WP4 components. We demonstrate the integrated behaviour of WP4 components and, on top of that, outline their interaction with other work packages, such as, WP5. Moreover, we show that the approach is not limited to the ACEA use case and generally applicable as shown in various published articles.

1.3 Scope

The scope of this deliverable includes published and unpublished work that is based on the components NIP, VIP, PM, MIM, LLC, and RMC.

1.4 Document Structure

Deliverable D4.3.2 is structured in the following manner:

Section 1 Introduction: Describes context, purpose and scope of the deliverable.

Section 2 Methodology: Describes methodology followed in the development of the deliverable.

Section 3 Data Collection: Presents scientific contributions belonging to RMC, VIP, NIP and the PM.

Section 4 Low-Level Correlation: Presents scientific contributions belonging to the LLC.

Section 5 Mission Impact Modeling and Mission Impact Assessment: Presents the scientific contributions belonging to MIM including the SNDA.

Section 6 Conclusion: Summarizes the findings, results and recommendations of this document.

2 METHODOLOGY

2.1 Selection of Test Data

The scientific work of WP4 and therefore the evaluation of the NIP, VIP, PM, MIM, LLC, and RMC components lead and still leads to paper contributions for different conferences. The following types of data were used for conducting experiments:

- *Synthetic Data:* in order to show some specific behavior of the components and their interaction, the experiments used direct control of the variation of specific parameters. To this aim, some evaluations have been done by generating synthetic data sets following specific patterns. Such data sets are however "'realistic"' in the sense that they have the same format and content as those coming from the real environment (e.g., IP addresses, devices etc.).
- *Real Data from the Emulation and Production Environment:* whenever possible, data coming from the Emulation or Production Environment was used to directly test the prototypes. When data coming from the Emulation/Production Environment could not be used directly to perform the whole experiment, they have been used to validate the outcome of tests done with synthetic data.
- *Hybrid Data:* In order to run specific tests, data coming from the Emulation Environment have been integrated with synthetic data.

Each paper contribution itself states how evaluation data was created and/or used.

2.2 Dissemination Activities

Approaches used in WP4 components have been published at major conferences, thus have been peer reviewed and have been discussed with practitioners. For the scope of the PANOPTESEC project, WP4 partners attended the following conferences.

- IJCAI 2015: 29th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July, 2015 (Appendix H).
- ICMLA 2015: 14th International Conference on Machine Learning and Applications, Miami, Florida, USA, December 9-11, 2015 (Appendix B).
- IST 2015: NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact, Istanbul, Turkey, June, 2015 (Appendix E).
- STIDS 2015: Tenth Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax, VA, USA, November, 2015 (Appendix A).

- ARES 2016: 11th International Conference on Availability, Reliability and Security, Salzburg, Austria, August, 2016 (Appendix F).
- CBI 2016: 18th IEEE Conference on Business Informatics, Paris, France, August, 2016 (Appendix G).
- CISIS 2016: 9th International Conference on Computational Intelligence in Security for Information Systems, 19–21st October, 2016 (Appendix C).
- KI2016: 39th edition of the German Conference on Artificial Intelligence, Klagenfurt (Austria), September 26-30, 2016 (Appendix D).

2.3 Quality Assurance

The quality assurance (QA) in the PANOPTESEC project relies on the assessment of a work product (i.e., deliverable) according to lists of QA checks (QA checklists) established by a QAM, validated at a consortium level and centralized in the project handbook. For the purpose of the QA of the D4.3.2 report, this deliverable has been peer reviewed by two independent reviewers, who completed the following documents:

- PEER REVIEW (PR) QA CHECKLIST: the D4.3.2 report deliverable is a report; hence there was a proper peer reviewed according to the checks defined in the accompanying checklist.

All comments made by reviewers have been incorporated into this D4.3.2 deliverable. The D4.3.2 deliverable has been revised based on the received remarks.

3 DATA COLLECTION

Within the context of the PANOPTESEC system, it is mandatory to be able to collect data from the monitored system in order to efficiently reconstruct the network topology and the vulnerability surface. The risk management system must be able to be updated with the most up-to-date information about monitored devices, their vulnerabilities and their reachability at ISO/OSI layer 3 and 4. In compliance with the non-functional requirement for the PANOPTESEC system, the system is designed to be modular in order to easier its maintainability and portability. For this reason, the reconstruction of the network topology is managed by several components, interacting among themselves with the PANOPTESEC Integration Framework (see Figure 3.1):

- Network Inventory Processor (NIP)
- Vulnerability Inventory Processor (VIP)
- Reachability Matrix Correlator (RMC)

A fourth component, the Persistency Manager (PM), is responsible for data persistency for all components of the PANOPTESEC System.

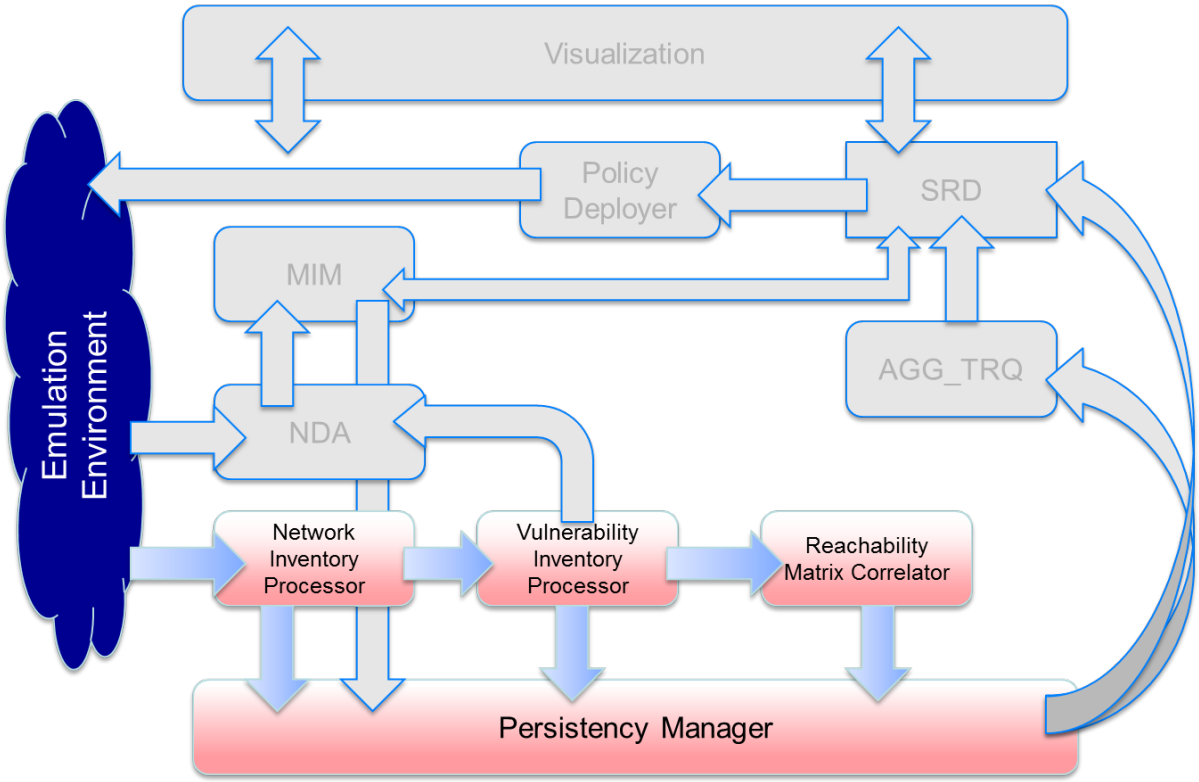


Figure 3.1: Overview of the PANOPTESEC Data Collection Process

3.1 Data Collection Related Data Model

The aim of the PANOPTESEC project is to build a system able to be adaptable to different networks (from ICT networks to critical SCADA networks) and to different data sources (topology scanners, inventory scanners, vulnerability scanners). In order to obtain this goal, a serious effort has been spent on the development of a common data model (a knowledge base) able to manage a variety of possible installations and data sources. One of the tasks of the Network Inventory Processor (NIP) and the Vulnerability Inventory Processor (VIP) is to collect raw topology/inventory/vulnerability data from the monitored system and to normalize them in the common data model which it is then used by the RMC component in order to correlate the reachability matrix at ISO/OSI level 3,4 (the overall result is a reconstruction of the network topology). The common data model is described in [D7.3.1] and managed in the SysML/UML PANOPTESEC design projects. Within the PANOPTESEC system all data are exchanged in Json format (corresponding Json schemes that have been produced and included in [D7.3.1]). The main products of the processing of the NIP and VIP are the network inventory (NI), the vulnerability inventory (VI) and the scored vulnerability inventory (SVI). RMC receives the NI and uses it (with the filtering policies of the monitored system) in order to compute the reachability matrix.

3.1.1 Network Inventory

The NI data model encompasses all network devices (ICT and SCADA) and useful information for the PANOPTESEC System, namely topology, inventory, vulnerability information. The *NetworkInventory* class is the base structure for most of the computations within the PANOPTESEC system, since it encompasses all monitored devices with all their retrievable information (see Figure 3.2). In particular, the *NetworkInventory* class has a unique identifier and it is tagged by a snapshot ID, which identifies the NI as part of a specific snapshot of the monitored system. The *NetworkNode* class encompasses all useful information for an active device in the monitored system. Each network node has a unique identifier which is kept through the lifetime of the PANOPTESEC system (so the NetworkNodeId will be always the same at every snapshot for the same device). In particular, the *NetworkInterfaces* property holds all network interfaces of the device (at ISO/OSI level 2,3 and 4). The *OperatingSystem* property encompasses all retrievable information about the operating system of the device, its installed applications, its users and installed/needed patches. Among the installed applications, it is identified if some of them are relative to a specific open port (for example FTP Servers, SSH, etc.).

3.1.2 Vulnerability Inventory

The knowledge representing the characteristics of elementary intrusion possibilities for devices of the monitored system is collected with a structured format: the VulnerabilityInventory format. This format basically describes vulnerabilities in an abstract way. For each vulnerability, the *requirements* that would allow their successful exploitation by an attacker, and the *consequences* produced by the successful exploitation of the vulnerability, are explicitly expressed (see Figure 3.3).

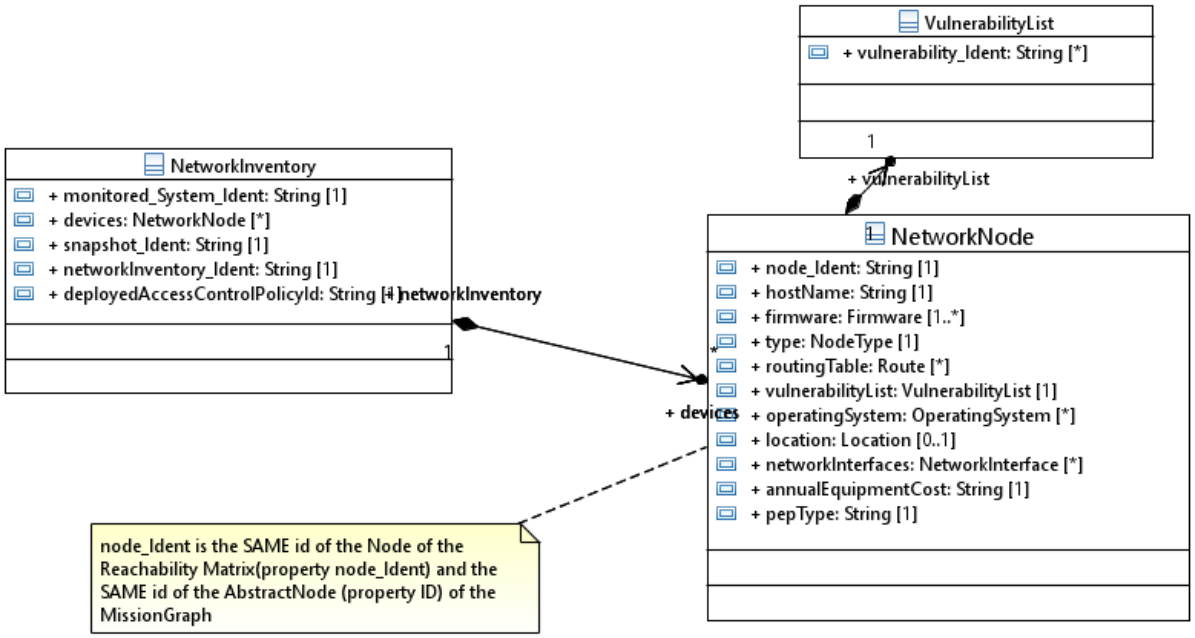


Figure 3.2: Network inventory data model

The *VulnerabilityInventory* class in Figure 3.3 defines a unique identifier and is tagged by a *snapshot ID*, which identifies a vulnerability inventory as part of a specific snapshot of the monitored system. The VulnerabilityInventory class holds a list of at least one instance of the Vulnerability class. The aggregate classes that compose *Vulnerability* are:

- *Reference*: At least one instance of the Reference class has to be provided. The Reference class provides all the identification knowledge of the considered vulnerability. Basically this class derives from the Reference class defined for the IDMEF format (cf. [RFC4765] Section 4.2.7.1), but with the purpose of describing a vulnerability and not an alert. This class is detailed below.
- *Requirement*: One instance of the *Requirement* class must be provided. This class structures the technical requirements that an attacker need to have fulfilled in order to have a chance to successfully exploit the considered vulnerability. Details about this class may be found below.
- *Consequences*: One instance of the *Consequences* class must be provided. It provides the technical gains of an attacker that successfully exploited the considered vulnerability. More details about this class are provided below.

The *Vulnerability* class is also composed of two basic attributes:

- *exploit_Exists*: is a real value, in the range $]0; 1[$, that indicates the likelihood (i.e. degree of confidence) of the existence of an exploitation method for the considered vulnerability.
- *patch_Category*: is a real value that provides a correspondence to the categorization of the patching method that may exist to address the considered vulnerability. This

correspondence makes a reference to a categorization of patching methods, defined within the entity that exploit the monitored or protected system to which the vulnerability inventory relates. This categorization and the corresponding values for the *patch_category* attribute should be defined consistently within the whole vulnerability inventory instance. But, the definition of the categorization and the corresponding values for this attribute is beyond the scope of the work package 4.

The Reference class is composed of five attributes:

- *vulnerability_ident* (mandatory): It is a character string that corresponds to a unique identifier of the considered vulnerability. For instance, the CVE [1] identifier could be used (e.g. CVE-2014-6278, CVE-2014-7186, CVE-2014-7187).
- *name* (optional): It is a character string that gives a human-readable identity to the considered vulnerability (e.g. "'Shellshock"').
- *url*: (optional); It is a character string representing a URL at which additional information can be found about the vulnerability. The pointed document may include an in-depth description of the vulnerability, appropriate fixes (e.g. patch to deploy, configuration to prevent the vulnerability exploitation), or other relevant information.
- *origin* (optional): It is a character string identifying the producer of the *vulnerability_ident* value or that identifies the *url* owner. The attribute has the same meaning and can take the same values as those defined in the IDMEF format for the origin attribute of the Reference class (cf. [RFC4765] Section 4.2.7.1).
- *description*: (optional); It is a character string that gives a human-readable description of the vulnerability. It may be, for instance, the overview given in the advisory of the NVD [NVD] for a vulnerability.

The Requirement class is composed of one attribute and two aggregate classes:

- *accessLevel* (mandatory): It represents the level of privilege an attacker requires to successfully be able to exploit the considered vulnerability. The attribute can take different values defined in the enumerated *Privileges*: "'none"', when no particular privilege is required; "'user"', when an unprivileged access to the machine is required; "'root"', when a privilege access to the machine is required.
- *ProtocolList*: an instance of the ProtocolList must be provided. It represents the list of protocols to which an attacker has to have access on the attacked device containing the considered vulnerability, to be able to exploit successfully the considered vulnerability on the device. The instance is linked to a list of instances of the *Protocol* class, which is defined in the *reachability matrix* section below.

[1] https://nvd.nist.gov/

- *PortList*: It represents the list of ports to which an attacker has to have access on the attacked device containing the considered vulnerability, to be able to exploit successfully the considered vulnerability on the device. The instance is linked to a list of instances of the PortRange class, which is defined in the *reachability matrix* section below.

The *Consequences* class is composed of the same attribute and two aggregate classes, but with a different meaning:

- *accessLevel* (mandatory): It represents the gained privilege of the attacker that successfully exploited the considered vulnerability. The attribute takes the various parameters defined in the enumerated Privileges.
- *ProtocolList*: An instance of the *ProtocolList* must be provided. It represents the list of protocols gained by the attacker that successfully exploited the considered vulnerability on the attacked device containing the considered vulnerability. The instance is linked to a list of instances of the *Protocol* class, which is defined in the *reachability matrix* section below.
- *PortList*: an instance of the PortList must be provided. It represents the list of ports gained by the attacker that successfully exploited the considered vulnerability on the attacked device containing the considered vulnerability.

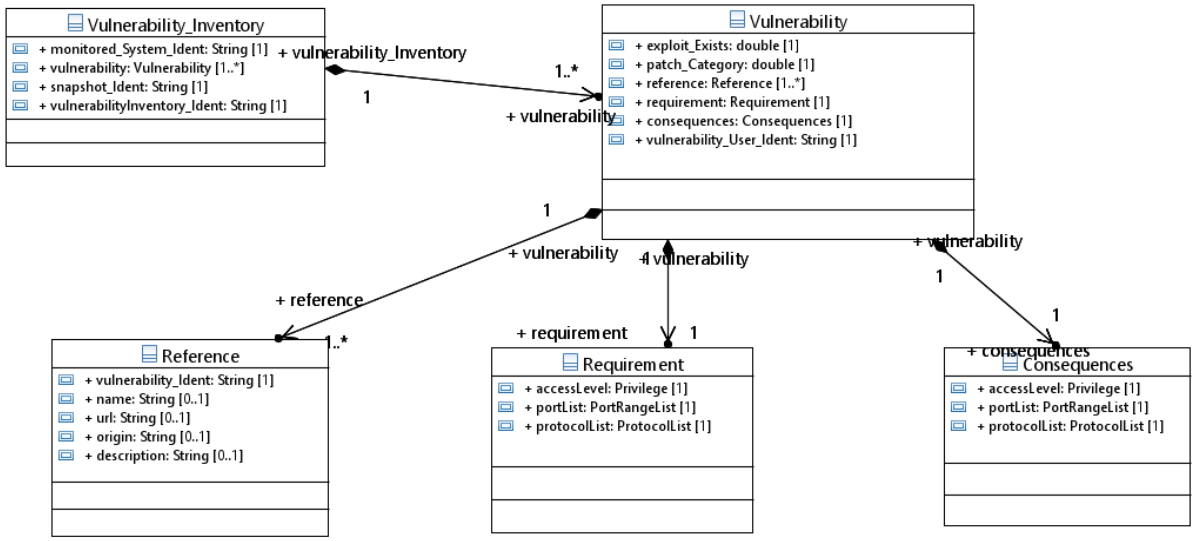


Figure 3.3: Vulnerability inventory

3.1.3 Scored Vulnerability Inventory

The scored vulnerability inventory provides the elementary vulnerability values provided by the [PKR2007] vulnerability scoring systems that are provided by the public US National Vulnerability Database.

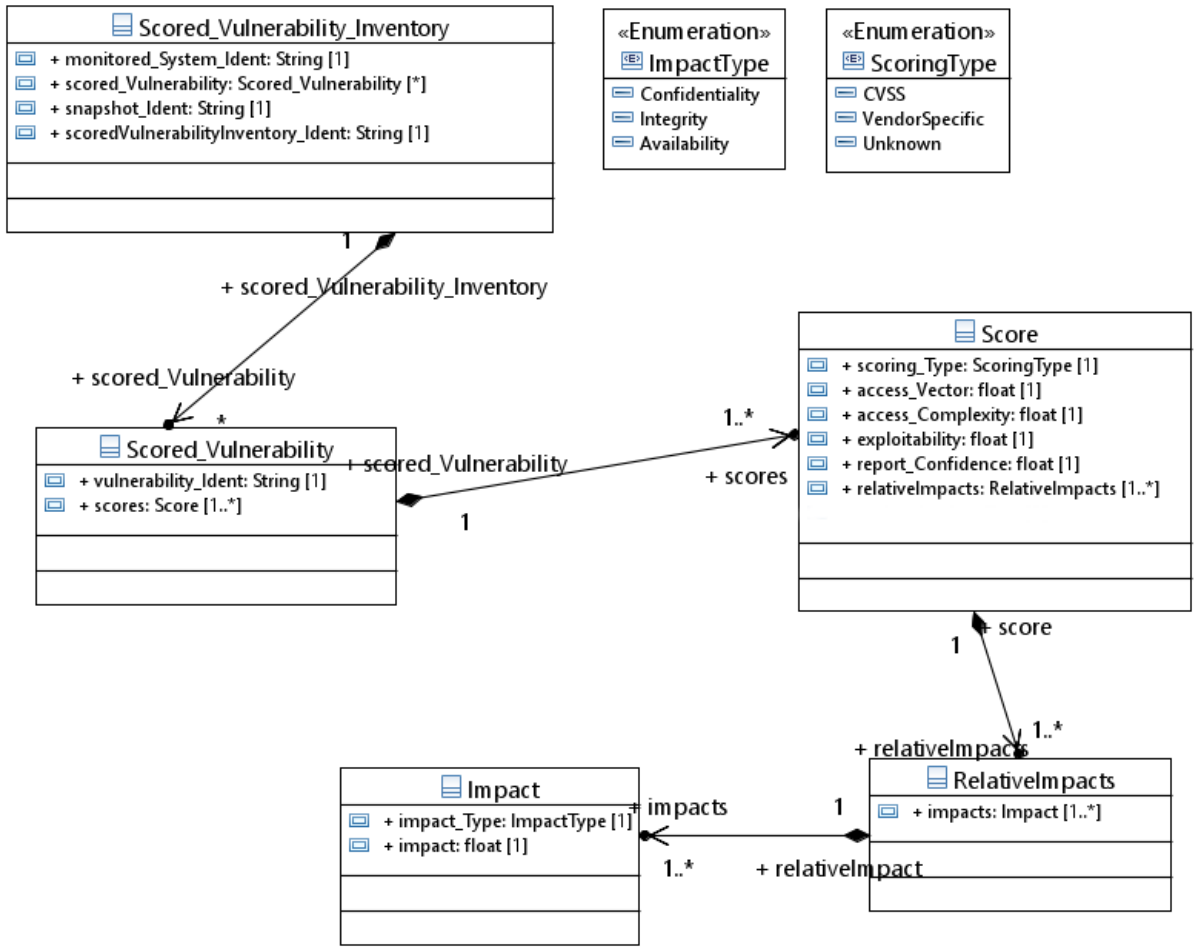


Figure 3.4: ScoredVulnerabilityInventory Data Model

The *ScoredVulnerabilityInventory* class in Figure 3.4 has a unique identifier and it is tagged by a *snapshot ID*, which identifies a ScoredVulnerabilityInventory as part of a specific snapshot of the MonitoredSystem. The *ScoredVulnerabilityInventory* class holds a list of *Scored_Vulnerability* instances. It is the including class that provides all the known scores for a given known vulnerability. The *Scored_Vulnerability* class is itself composed of one attribute and an aggregated class:

- *vulnerability_ident* (mandatory): It is a character string that relates to the unique identifier provided in the related Vulnerability Inventory of the considered monitored or protected system.
- *Scores*: One or several occurrences can be provided. It provides the basic score values from a specific source for the vulnerability identified in the including Scored_Vulnerability instance.

The Score class is composed of several attributes:

- *scoring_type* (mandatory): The attribute can take one of the values defined in the enumerated SoringType: "'CVSS"', if the values are extracted from a CVSS public database;

"'VendorSpecific"', is the score values are provided by the organization managing the PANOPTESEC system itself; "'unknown"', otherwise.

- *access_vector* (optional): It provides the real value corresponding to an access vector metric (AV) of a CVSS Based Score, in the range $]0;1[$, as specified in the [PKR2007] scoring system.
- *access_complexity* (optional): It provides the real value corresponding to an access complexity metric (AC) of a CVSS Base Score, in the range $]0;1[$, as specified in the [PKR2007] scoring system.
- *exploitability* (optional): It gives the real values corresponding to an exploitability metric (E) of a CVSS Temporal Score, in the range $]0;1[$, as specified in the [PKR2007] scoring system.
- *report_confidence* (optional): It provides the real value corresponding to a report confidence metric (RC) of a CVSS Temporal Score, in the range $]0;1[$, as specified in the [PKR2007] scoring system.

The score class is also composed of an aggregated class:

- *RelativeImpacts*: one instance must be provided. This is an including class that provides the list of the various impacts of the considered vulnerability. It is further composed of one or several instances of the Impact class. Each instance of the Impact class gives the impact magnitude and its nature on defined security axis (i.e. Confidentiality, Integrity, and Availability). The Impact class is composed of
  - *impact_type* (mandatory): It gives the nature of the considered impact potentially produced by the exploitation of the vulnerability on a device on which the vulnerability exists. Currently, the attribute can take three values defined in the enumerated *ImpactTypes* class. Note that other security axes could be envisaged (e.g. the eight dimensions of the NIST X805 standard), but this implied that the T*hreat Risk Quantificator* implementation is able to address them appropriately within its internal processes.
  - *impact* (mandatory): It is a real number that gives the magnitude of the impact of exploiting the considered vulnerability of a device for the dimension provided in the *impact_type* attribute.

3.1.4 Reachability Matrix

In order to provide to the rest of the PANOPTESEC system knowledge about communication possibilities (i.e. at a logical network level) between devices of monitored or protected systems for which the possible attack graph has to be generated, a specific representation is defined as an instance of the class: *ReachabilityMatrix*. This representation provides an exhaustive view of which device can communicate with each other devices of a network topology, up to the

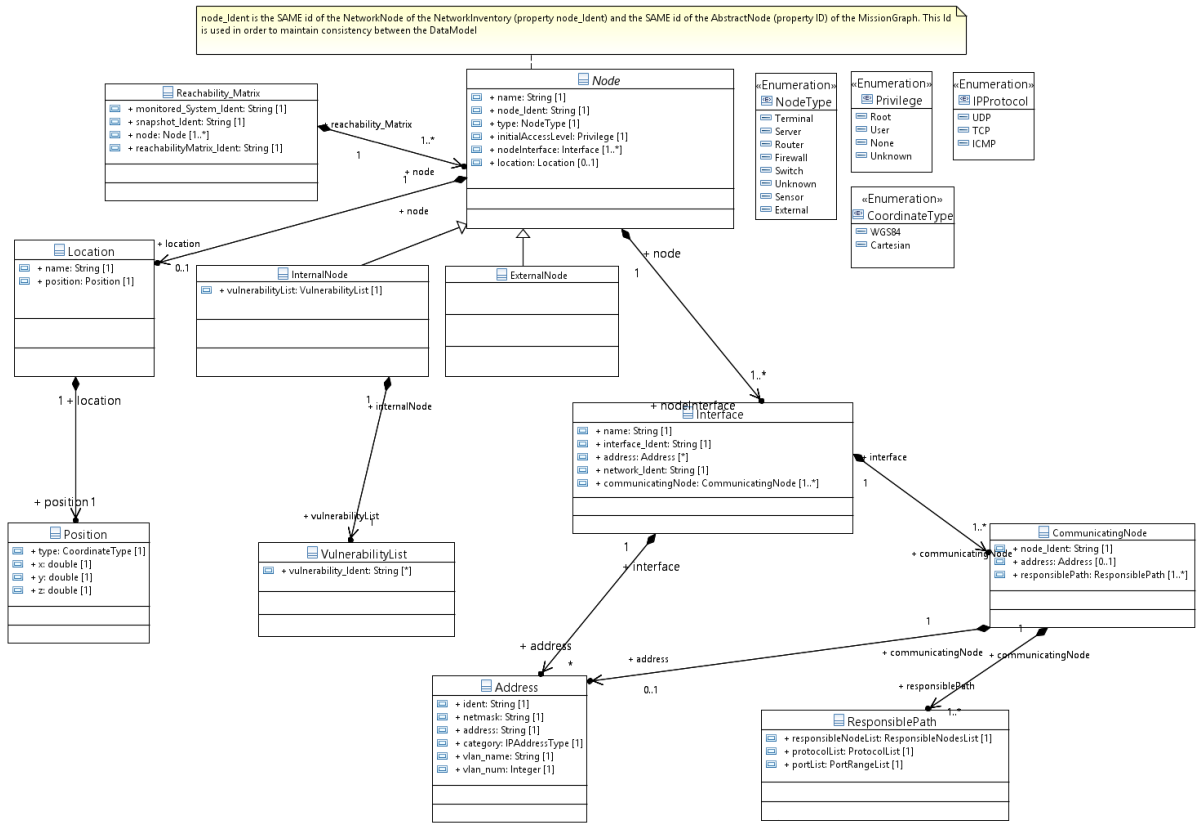


Figure 3.5: ReachabilityMatrix Data Model

ISO/OSI level 4. The knowledge should not only provide a view of the logical routing possibilities, but also encompass the knowledge of the filtering policy currently deployed in the topology (i.e., deployed firewalling rules in every device of the topology). Basically, for an IP network topology, the reachability matrix contains information that could be represented by a two dimension matrix or table, each column and row being IP addresses. In this matrix or table, each cell carries the information of the IP ports and protocols authorized between the coupl respectivee of IP addresses, in one direction and the other. Additionally, the representation also carries the vulnerability state of each device of the monitored or protected system it represents. The *ReachabilityMatrix* class in Figure 3.5 defines a unique identifier, and it is tagged by a *snapshot ID*, which identifies a reachability matrix as part of a specific snapshot of the monitored system. The *ReachabilityMatrix* class holds a list of *Node* instances, representing single devices in the monitored system. Each *Node* instance carries a unique identifier, *node_ident* (the same unique identifier of the class *NetworkNode* from the network inventory class diagramm).

Node is a class composed of three other basic kinds of attributes:

- *name* (mandatory): Contains a character string representing the device in the topology of the monitored or protected system for a human (e.g. Security Officers, Administrators, etc.).
- *type* (mandatory): Defines the type of the device potentially corresponding to network features of the device in the topology of the monitored or protected system (e.g. a

firewall, router, server or simply a terminal node). It may take at least one of the values defined in the enumerated *NodeTypes*. Note that a device can be of several types as it may have several functions in a network.

- *InitialAccessLevel* (mandatory): Indicates the initial level of privilege on the device that a user of the monitored or protected system may obtain without exploiting one of the vulnerabilities existing on the device. Usually, when a protocol or port is accessible on a device, it is considered that a user has an unprivileged access to the service replying on those ports or protocols, in that case we consider that *InitialAccessLevel* attribute takes the 'user' value.

The Node class is further decomposed in three classes representing different kinds of relevant knowledge:

- *Location*: a class that carries the identification and physical location (e.g., the building) of the device. One or several instances of this class can be attached to a *Node* instance. It is decomposed into two parts:
  - *name* (mandatory): A character string representing the location.
  - *Position*: At least one instance of the Position class, defining geographical information of the location. The Position class is further decomposed in
    * *type*: The coordinate system used. Currently, it can take two values, defined in the enumerated *CoordinateTypes*.
    * *x, y, and z*: Are three attributes storing the position of the location following the coordinate system expressed in the type attribute.
- *VulnerabilityList*: A class that gives the list of vulnerabilities existing on the considered device. One instance of this class must be provided for each instance of the *Node* class. This class defines one kind of attribute:
  - *vulnerability_ident*: A character sting that represents the unique identifier of a vulnerability as identified in the vulnerability inventory corresponding to the monitored or protected system.
- *Interface*: a class that carries the network connection and communication information of the device. Several instance of that class may be provided in each instance of the *Node* class. This class is composed of several attributes:
  - *name* (mandatory): A character string representing the considered network interface.
  - *Interface_ident* (mandatory): For a unique identifier that identifies the considered network interface.
  - *network_ident* (mandatory): A unique identifier for the network to which the considered interface is attached.

  - *address*: Exactly one occurrence of the Address class is mandatory. The *address* follows exactly the same definition as the one defined in the IDMEF format (cf. [RFC4765] Section 4.2.7.2.1).
  - *communicatingNode*: At least one occurrence of the *CommunicatingNode* must be provided.

The *CommunicatingNode* class is the main entity that carries the important connection and communication information in a Reachability Matrix. It basically expresses, for each protocol or port, through which routing path another node can reach the node identified in the *CommunicationNode* instance; a routing path being expressed with a *ResponsiblePath* instance. *CommunicatingNode* has two mandatory attributes that identifies the considered node (i.e. *node_ident* and its *interface_ident*, identifying the entering interface), target of the communication capabilities expressed in at least one instance of the *ResponsiblePath* class.

The *ResponsiblePath* instance is further decomposed in (not shown in Figure 3.5):

- *ResponsibleNodeList*: One or zero instance of this class can be provided. It expresses the list of successive nodes (i.e. responsible nodes) that enable issuing of IP packets to the identified node in the linked *CommunicatingNode* instance. This class is composed of a list attribute, *node_ident_list[*]*, which stores the ordered list of the successive nodes through which the considered IP packets goes.
- *ProtocolList*: Represents the list of protocols that are authorized with the considered node (i.e. the node identified in the *CommunicatingNode* instance to which the *ResponsiblePath* instance is linked). The protocols of the list are expressed using instances of the *Protocols* class.
- *PortList*: For the protocols that cannot be represented by an abstract protocol name, it represents the list of ports that are authorized with the considered node through the *ResponsiblePath*. The ports are expressed using instances of the *PortRange* class.

When an abstract name is given to a protocol in the IP domain, it is usually identified with a string composed of a name and a version (e.g. HTTP/1.1, SSHv2 etc.). The Protocol class is then composed of:

- *name*: An attribute, that is a character string representing the protocol (e.g., "'https"', "'ftp"', "'smtp"' etc.).
- *version*: An attribute which is a character string representing the version of the considered protocol.

The *PortRange* class enables expression of any kind of range of port, for any kind of IP protocols. The class is composed of:

- *ip_protocol* (mandatory): The used protocol (e.g., TCP or UDP).

- *port_list* (mandatory): A port range is expressed with a character string.

The *port_list* attribute value, in the *PortRange* class, must use the following pattern format:

```
[0-9]{1,5}([0-9]{1,5})?
```

Note that, the first part and last part of the pattern (i.e. '[0-9]{1,5}') have to be a number in the range [0;65535]; and the first number in the pattern has to be lower than the second number. For instance, the following strings are all valid port ranges: `'80'`, `'0-1023'`, `'1024-65535'`, `'6000-6010'`, `'25'`.

3.2 Network Inventory Processor

NIP is the first component within the proactive chain of the PANOPTESEC system (cf. [D3.1.2]). Its purpose is to analyze and monitor topology/inventory data sources, detect changes in the monitored system, normalize the data from the data sources in order to compute the NetworkInventory, maintain the up-to-date knowledge (within the Persistency Manager) of all devices of the monitored system. In addition, NIP is responsible for starting a new iteration (a new snapshot) of the proactive chain (since it is able to detect changes in the monitored system which may trigger updates concerning attack paths and risk assessment). In Figure 3.6, the final high level design of the network inventory processor is shown. The aim of NIP is to connect

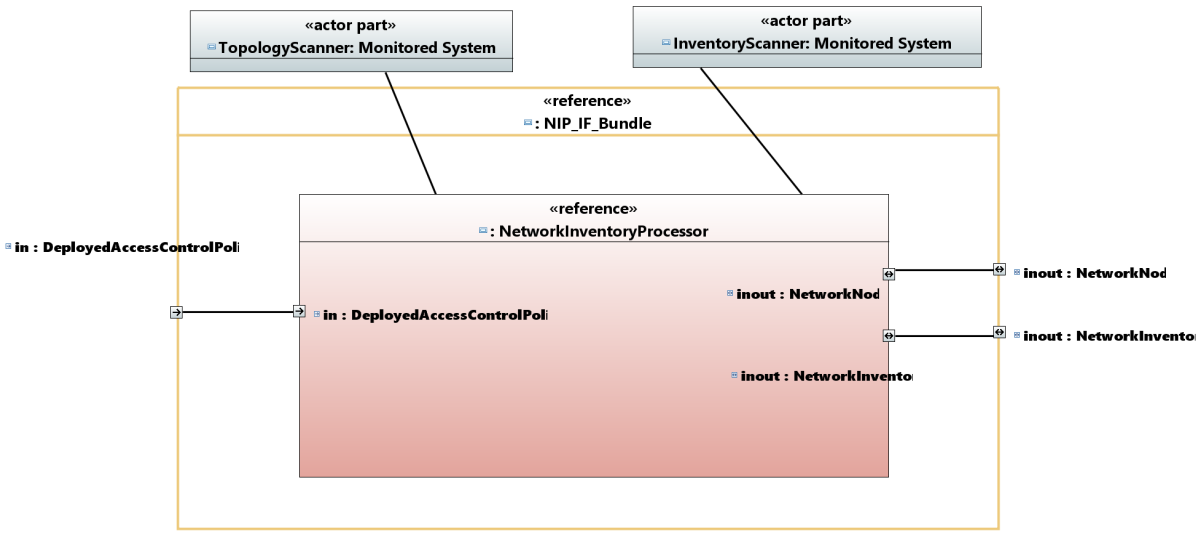


Figure 3.6: NetworkInventoryProcessor high-level design

with all available data sources (topology scanners, inventory scanners) and orchestrate the normalization process in order to retrieve any useful information from each of them. Since the network inventory data model can be used in order to model a variety of different devices, no matter for which data source respective information is extracted. NIP is built around a generic normalization core able to use specific data source readers, added at component start-up as plug-ins for NIP. It is possible to develop a specific data source reader and add it to NIP as a

new plug-in if a new data source must be added to the PANOPTESEC System. NIP will be then able to use the reader (hence the new data source) during its data collection and normalization processes.

NIP is thus able to combine the results of multiple different data sources in order to enrich the produced NetworkInventory with more information.

Among all data sources, one is defined as primary, for example, an inventory scanner, giving inventory and topology information: at each iteration NIP checks the data sources starting from the primary source in order to verify if a new set of data is available. If anything in the monitored system underwent a change since the last check, NIP collects all data from the sources and generates a new version of the network inventory.

The following changes are possible in the monitored system managed by NIP:

- a device is added/removed
- a firewall rule on a monitored router/firewall is changed
- a patch is installed for an application (or operating system) on a monitored device
- a port is opened/closed on a monitored device
- routing rules on a monitored device are changed
- a network interface is added/removed on a monitored device
- a new application is installed/removed on a monitored device
- a vulnerability is detected as added/removed on a monitored device

NIP can be configured in order to reduce the sensitivity over the perceived changes (for example, it is possible to ignore one of the above mentioned triggers).

NIP manages each data source with a different plug-in connector (configurable at startup). Each connector manages the interactions with the data source and the orchestration of the information retrieval once the main trigger within the NIP core elapses. This architecture allows NIP to benefit of each single integrated data source while remaining agnostic with regard to the external data collection process.

NIP can be configured for managing conflicts between different data sources (for example it is possible that different topology scanners identify different topology properties on the same device). The default configuration of NIP gives priority to the most recent information.

In the current installation of the PANOPTESEC System within the ACEA environment, four topology data sources are used, covering all monitored systems:

- GFI Languard (topology/inventory/vulnerability scanner, mainly used in order to retrieve inventory information)

- Whatsup Gold (used for checking whether a network interface is active or not)
- NetworkInventoryProcessorAgent, (NIPA, developed within the PANOPTESEC Project, a fast topology scanner based on NMAP (Network Mapper) allowing NIP to retrieve extremely fresh information)
- OpenVAS (Open Vulnerability Assesment System) a vulnerability scanner. Used for verifying updates on the vulnerability set of each monitored device.

NIP does not make vulnerability analyses – which is a task for VIP, but simply verifies if a device has a new vulnerability or whether a vulnerability it not present anymore.

NIP can work with a configurable set of data sources. The accuracy of results, of course, will depend on the accuracy of the sources. For example, it is possible to configure NIP for working with only the network inventory processor agent (of course, the only information which is available will be topology information, sufficient, however, to reconstruct the network topology and trigger the proactive chain). NIP keeps in the Persistency Manager the up-to-date list of discovered devices, each of it identified by a unique identifier. The computed network inventory, however, lists only the active devices (devices with at least one active network interface). While not a default configuration, NIP can use different data sources for different segments of the network. For example, it is possible to use two different scanners monitoring two different sets of broadcast domains. This behavior is useful when some of the scanners have a license based on the number of monitored devices. In the case of the production environment of ACEA distribution, with around 1000 medium voltage substations available for active scanning, this is a viable solution. The command and control core network is monitored also by the inventory scanner (GFI Languard), while the substations are monitored by NIPA only in order to reduce the license costs coming with Languard. The unique ids given by NIP every time a new node is scanned are then used by all subsequent components of the PANOPTESEC system in order to identify devices in the monitored system within all correlated data. NIP also manages the creation of a snapshot, which is a foundation concept for the proactive chain of the PANOPTESEC system. Each scan of the monitored system, and the subsequent processes of normalization and creation of the network inventory, also creates a new snapshot (kept in the PANOPTESEC persistency manager), with an unique id and a timestamp. For each computed dataset needed for the proactive chain (*VulnerabilityInventory*, *ScoredVulnerabilityInventory*, *ReachabilityMatrix*, *MissionGraph*, *ShallowNetworkDependencyModel*, *AttackGraph*, *EnrichedAttackGraph*, *ProactiveRiskProfile* and the possible set of proactive *ResponsePlans*) consistency is maintained across computations via a snapshot ID. This means that NIP is the driver for the proactive chain, while the persistency manager has the task to evaluate the current status of the chain. Information retrieved by NIP can be enriched with data inserted by humans (for example for automatic scans, it is in some cases too complex to decide whether a certain device has multiple interfaces).

3.3 Vulnerability Inventory Processor

The vulnerability inventory processor (VIP) component is responsible for the analysis of the vulnerabilities of devices collected by the network inventory and manages the computation of the *VulnerabilityInventory* and *ScoredVulnerabilityInventory* instances. Vulnerabilities identified by scanners are gathered, and their properties are identified by querying the CVE database created by the Vulnerabilities Advisory Collector (a PANOPTESEC system component responsible for regularly downloading and updating the local NVD database, see subsection 3.3.1). With this information, vulnerability inventory and scored vulnerability inventory data are generated and sent to the Persistency Manager. The network inventory is also enriched with vulnerability advisory information for nodes identified to have known vulnerabilities. In order to compute

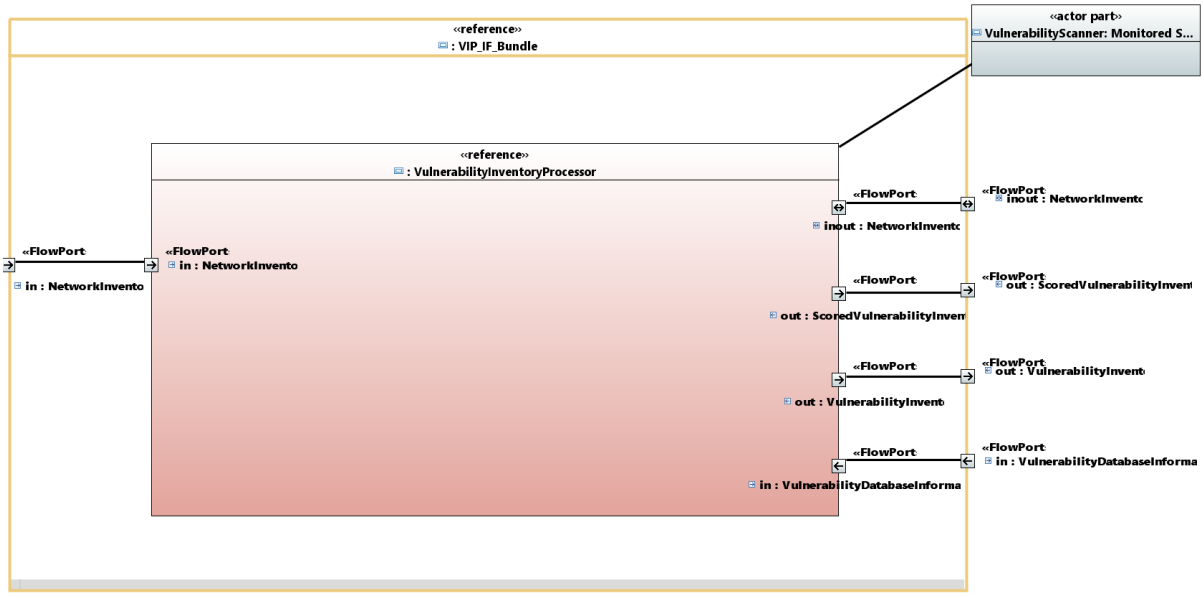


Figure 3.7: VulnerabilityInventoryProcessor High Level Design

the vulnerability inventory and the scored vulnerability inventory, VIP must analyze several data sources and combine respective information.

3.3.1 Vulnerability Information Data Sources

Vulnerability management of ICT systems routinely makes use of dedicated database repositories maintained by government and commercial enterprises providing vulnerability information to interested partners. Such services may be offered free of charge, but can also be available under payment, or subscription based agreements. The National Vulnerability Database (NVD) maintained by the U.S. government is perhaps the best known of these, providing users with a continuously updated list of Common Vulnerabilities and Exposures (CVEs), discovered and documented for an extensive list of configurations for hardware platforms, operating systems and software applications. To date, this database contains over 70000 known distinct vulnerabilities (each with a unique CVE index), submitted by vendors, IT security experts, researchers, and interested individuals, dating as far back as 1999. A vulnerability listed by NVD is identified by a

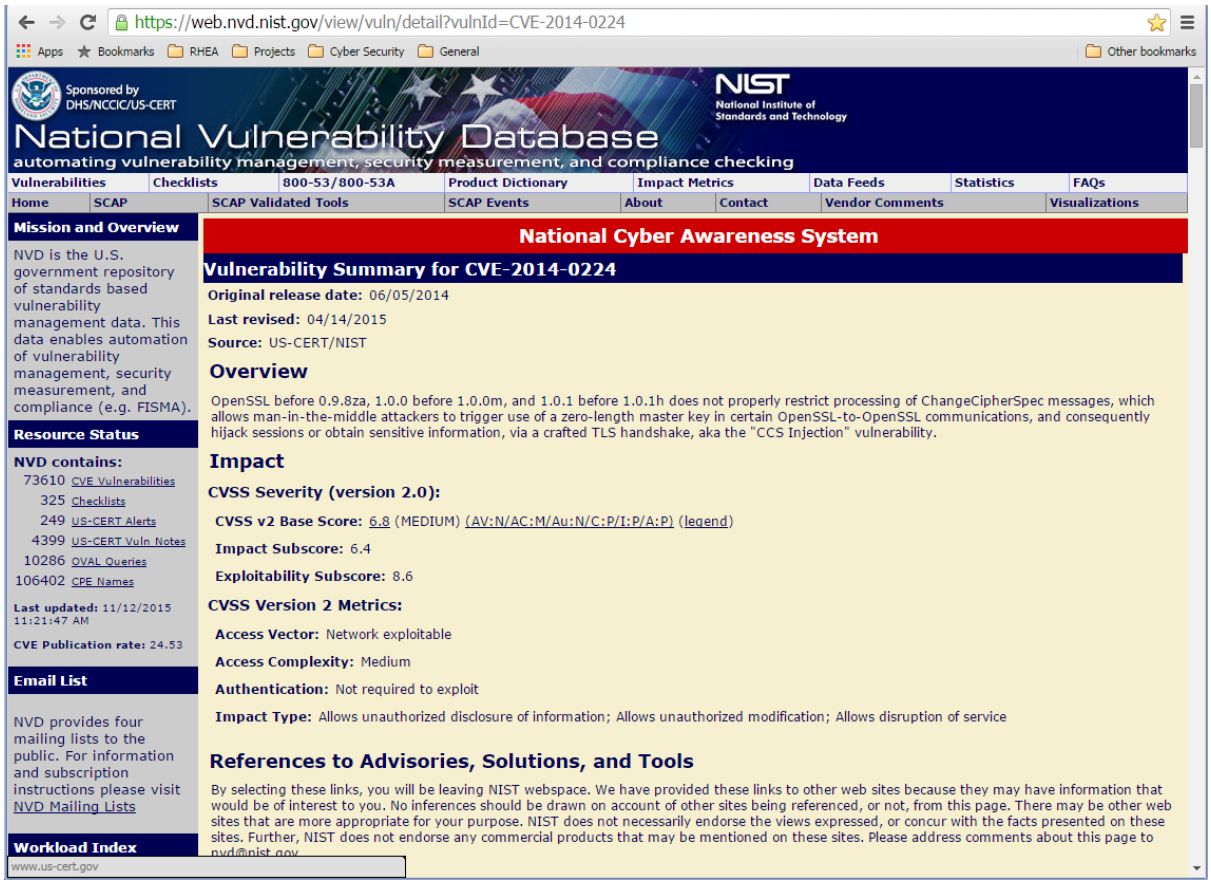


Figure 3.8: National Vulnerability Database

unique CVE identifier assigned when the vulnerability is first discovered, submitted and verified by expert staff at the Computer Emergency Response Team Coordination Center (CERT-CC). Information about the vulnerability (including impact metrics, security checklists, software flaws, and misconfigurations) and a quantified measure of the potential exploitability are stored in XML format to facilitate ingestion and processing through application programming interfaces (APIs). Currently, the Common Vulnerability Scoring System (CVSS) v2 is employed to provide the scoring standard. Vulnerability information can be downloaded from NVD from data feeds, or can be queried online using the NVD vulnerability search engine. An example of an online search for a specific vulnerability is given in Figure 3.8. An edited selection of the content of the XML data feed for the same vulnerability is shown in Figure 3.9. Alternatives to NVD vulnerabilities exist, but are in general contained in backend databases for open source vulnerability management systems (such as OpenVAS), or are given by professional commercial vulnerability scanners (such as LanGuard from GFI, Nessus from Tenable, Symantec Endpoint Protection and Secunia's Vulnerability Intelligence Manager). The information available from all of these sources is often complementary and it is sometimes necessary to combine data from multiple products to extract certain details of a known vulnerability, which may be missing from a single source. This also allows for vulnerability details to be corroborated using different sources, an important aspect given the dynamic nature of vulnerability detection and notification.

```
<entry id="CVE-2014-0224">
  <vuln:vulnerable-configuration id="http://www.nist.gov/">
    <cpe-lang:logical-test operator="OR" negate="false">
...
  <vuln:vulnerable-software-list>
    <vuln:product>cpe:/a:openssl:openssl:0.9.8i</vuln:product>
...
    <vuln:cve-id>CVE-2014-0224</vuln:cve-id>
  <vuln:published-datetime>2014-06-05T17:55:07.817-04:00</vuln:published-datetime>
  <vuln:last-modified-datetime>2015-04-14T22:00:10.777-04:00</vuln:last-modified-datetime>
  <vuln:cvss>
    <cvss:base_metrics>
      <cvss:score>6.8</cvss:score>
      <cvss:access-vector>NETWORK</cvss:access-vector>
      <cvss:access-complexity>MEDIUM</cvss:access-complexity>
      <cvss:authentication>NONE</cvss:authentication>
      <cvss:confidentiality-impact>PARTIAL</cvss:confidentiality-impact>
      <cvss:integrity-impact>PARTIAL</cvss:integrity-impact>
      <cvss:availability-impact>PARTIAL</cvss:availability-impact>
      <cvss:source>http://nvd.nist.gov</cvss:source>
      <cvss:generated-on-datetime>2014-06-06T10:29:47.773-04:00</cvss:generated-on-datetime>
    </cvss:base_metrics>
  </vuln:cvss>
  <vuln:cwe id="CWE-310"/>
...
  <vuln:references xml:lang="en" reference_type="PATCH">
    <vuln:source>CONFIRM</vuln:source>
    <vuln:reference href="
    https://git.openssl.org/gitweb/?p=openssl.git;a=commit;h=bc8923b1ec9c467755cd86f7848c50ee8812e441" xml:lang="en">
    https://git.openssl.org/gitweb/?p=openssl.git;a=commit;h=bc8923b1ec9c467755cd86f7848c50ee8812e441</vuln:reference>
  </vuln:references>
...
  <vuln:summary>OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict
  processing of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero-length
  master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain sensitive
  information, via a crafted TLS handshake, aka the "CCS Injection" vulnerability.</vuln:summary>
</entry>
```

Figure 3.9: Vulnerability description from NVD example

3.3.2 Patch Management Information

Patch management is an example of information related to vulnerabilities which requires special attention. The NVD data feeds provide limited information about how to remediate discovered vulnerabilities. If such information is indeed available, it is indicated by a flag in the list of public references provided in the data feed for the vulnerability. Often, this is accomplished in the form of an advisory or a set of patch instructions supplied directly by the vendor to address the vulnerability in question. In the absence of necessary information from the vendor, a third party may supply details to allow the vulnerability to be successfully resolved. The type of reference information provided by the NVD feeds for a patch is varying. In general, a single link to a URL for downloading an executable file that can initiate the remediation procedure is not available. In most cases, the reference will take the user to a vendor site where a detailed list of instructions may be found to patch the vulnerability. The Microsoft Knowledge Base (KB) repository is a prime example of such an approach. However, the complexity of the instructions may be such that only manual intervention by an experienced system operator can successfully remediate the problem. For example, it may be necessary to edit the computer registry or to enable or disable certain services in a specific order. For Microsoft products, patch management may be facilitated using the Windows Server Update Services (WSUS), commonly employed by corporate system administrators to manage the distribution and implementation of updates and hotfixes across corporate networks. For UNIX and Linux systems, patch bundles are

deployed by the Package Manager which executes crafted scripts. Examples of such programs are Red Hat Package Manager (RPM), Yellowdog Updater (YUM) and Advanced Packaging Tool (APT). Vulnerability scanners, such as LanGuard, combine information from WSUS and Package Managers to provide automated patch management for discovered vulnerabilities. Accessing relevant backend databases of LanGuard can provide access to specific patch information if needed.

3.3.3 Port and Protocol Information

Information pertaining to open ports and their protocols is also not readily available from the vulnerability data feeds provided by NVD. Such detail can possibly be extracted from careful assessment of the vulnerability description which may specifically identify ports targeted by the vulnerability. The LanGuard scanner provides a complete list of all open ports on a particular node. A discovered vulnerability will in general only be susceptible to a single open port or a subset of all open ports. It is important to be able to identify only relevant ports for a specific vulnerability in order to be able to assess how compromised nodes can affect neighbouring nodes. Fortunately, the OpenVAS vulnerability management system can associate specific ports with specific vulnerabilities using plugins that query a list of scripts used by OpenVAS to test open ports previously identified by NMAP (a free cross-platform network mapper). The OpenVAS plugins can also be queried online, as shown in Figure 3.10. The analysis over ports related to vulnerabilities is a main task of the Vulnerability Inventory Processor, because it is a fundamental information for the computation of the attack paths (in combination, of course, with the reachability information over ports and IPs which is given by the Reachability Matrix).

3.3.4 VIP Implementation for Patches and Ports/Protocols

The implementation of the VIP component for the PANOPTESEC project accesses a local store of the NVD data feed. This feed is collected on a regular basis by the Vulnerability Advisory Collector component, which downloads the most recent copy of the NVD data feeds in XML, converts them to JSON objects and finally stores them in a PostgreSQL database used by the PANOPTESEC Persistency Manager. Metrics and scores from the NVD data database are also stored by the PANOPTESEC Persistency Manager. They are used to construct the basis for building the *VulnerabilityInventory* and *ScoredVulnerabilityInventory* instances. Vulnerabilities are discovered for PANOPTESEC nodes by employing the LanGuard vulnerability scanner, which provides a list of CVEs found for alerts from installed software applications and, in addition, missing operating system patches. The NVD data feeds provide an indication of whether a patch is available for a particular vulnerability. In the case of a vulnerability arising from a missing patch, a URL pointing to a fix (usually an executable file or a cabinet file) is also identified for inclusion in the network inventory file. A secondary vulnerability scanner (OpenVAS) is employed to complement the information collected by LanGuard concerning ports and protocols. As stated above, Languard can only list all open ports on a particular node, but is unable to associate these ports with specific vulnerabilities and their CVE data feeds. OpenVAS is used to collect an independent list of CVEs. When a vulnerability has been identified by both LanGuard and

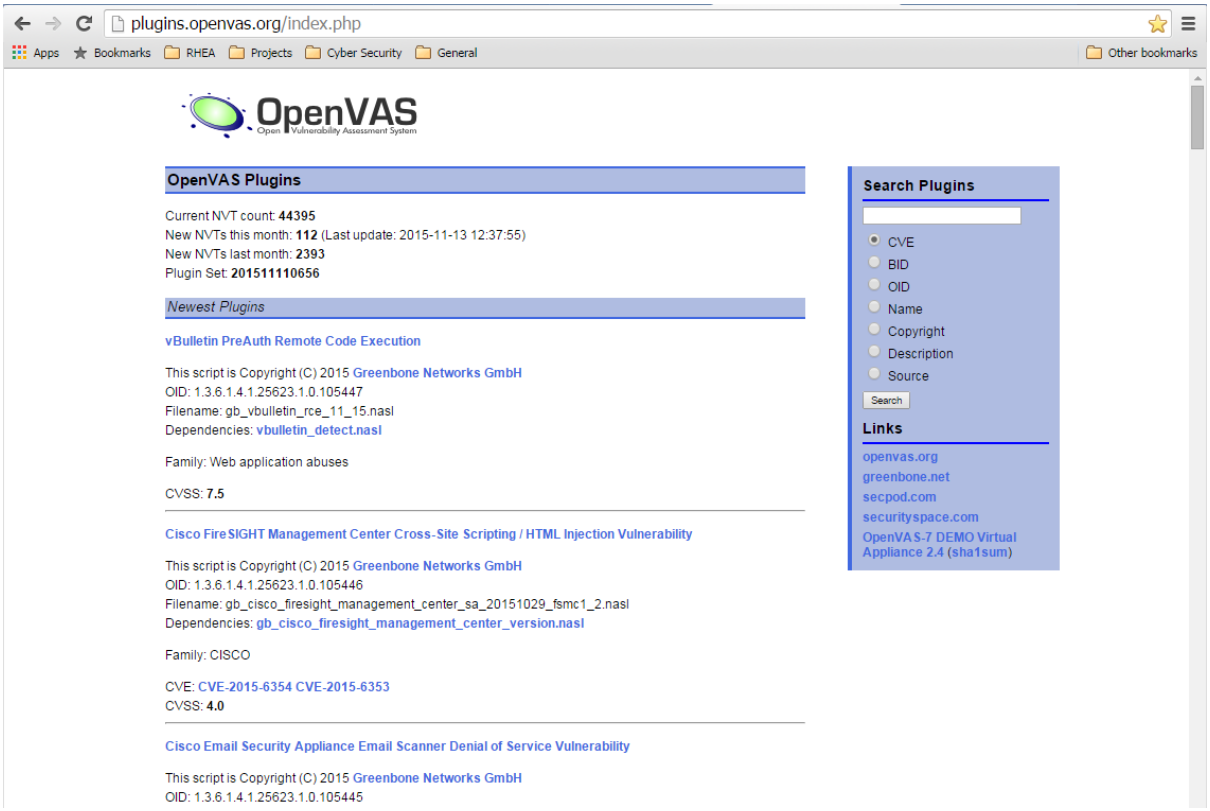


Figure 3.10: OpenVAS vulnerability scanner

OpenVAS, port and protocol information added to the VI by LanGuard is superseded by that from OpenVAS. If OpenVAS discovers a new vulnerability (with a CVE index not found by LanGuard), the VIP first checks if the open port (or ports) identified by OpenVAS is included in the complete list of open ports found by LanGuard. If it is, the new CVE is added to the vulnerability list contained in the (S)VI. If, on the other hand, there is no match between the OpenVAS port(s) and the LanGuard, the vulnerability identified by OpenVAS is ignored and excluded from the (S)VI. VIP is developed in synchronization with NIP. It is possible to build plug-ins in order to manage a new and different data source, while the VIP core remains completely agnostic about the source of the data which it manages.

3.4 Reachability Matrix Correlator

The Reachability Matrix Correlator (RMC) provides the reachability matrix, useful for the attack graph generation component of the Dynamic Risk Management Response System. This correlator component performs the reachability computation across the monitored ICT network to determine whether two nodes are reachable from each other in the network, and this for all pairs of nodes representing ICT devices. RMC utilizes a novel, ontology based approach to combine information from various sources to obtain a reachability matrix, which is required by various components, such as the attack graph generator of work package 5. The RMC ontology provides a deep description of the data model of PANOPTESEC , with special regards for the schemas concerning the network inventory and the deployed access policies, and the schema

for the reachability matrix . The RMC exploits SWRL (Semantic Web Rule Language) rules. These rules describe all possible scenarios to be investigated in order to detect all nodes that are reachable from any given pair of node (with its specific routing instructions) and associated network interfaces (each being connected to one particular network). RMC makes use of the RACER reasoner. The execution of reasoning based on all information within the knowledge base and the SWRL rules, allows for producing the set of all pairs made of a network interface and the nodes it can reach. Results of reasoning are refined through SPARQL queries in order to produce the Reachability Matrix. The ontology based approach was presented and peer-reviewed at the 10th conference on semantic technologies for intelligence, defense and security (STIDS 2015), held during November 2015 in Fairfax, Virginia, USA. The corresponding paper is given shortened in Appendix A.

3.5 Integration of Components and Evaluation of Performance

During the development of the PANOPTESEC system, the data collection components have been integrated within the integration framework after the first development iteration, and every new version has been reintegrated and tested within the context of the others. Although logically decomposed into multiple components, their aim is to provide an overall reconstruction of the network and vulnerability topology to the subsequent modules of the PANOPTESEC system.

The integration framework is described in D3.1.2, and it has been developed by WP7 with a technological stack based on Apache Servicemix, Apache Camel, Apache ActiveMQ as well as Apache CXF. Figure 3.1 encompasses a high level logical view of the components integrated into the data collection subsystem of PANOPTESEC. In terms of interactions between components the integration of the components follows the low-level design of the PANOPTESEC system as described in D7.4.2.

Each single component part of the data collection component has been verified with respect to the functional and non-functional requirements described in D4.1.1. The results of the verification phase have been presented in D4.3.1. The evaluation of single components has been conducted using synthetic data and data from the emulation environment, with the aim of assessing if the computed results were in line with the specifications. As soon as the first integration phase has been concluded, the data collection has been tested as a whole, in order to assess performances, reliability and stability of the set of components. Tests over the integrated data collection components have been performed using data from the emulation environment within ACEA Distribuzione, while scalability have been tested with data from the emulation environment enriched with data from the production environment within ACEA Distribuzione.

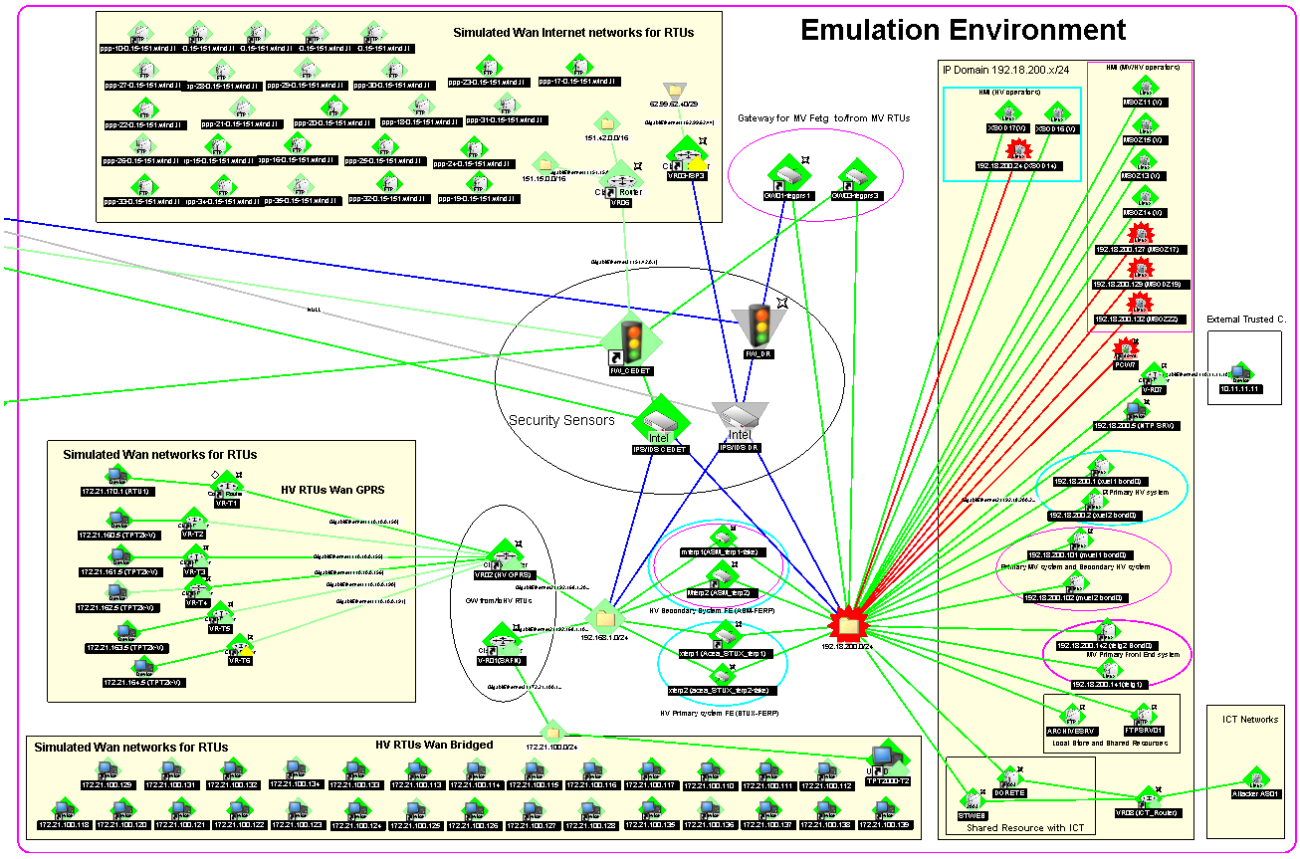


Figure 3.11: Logical view of the emulation environment.

In Figure 3.11, a logical view of the emulation environment from the Whatsup Gold Network Management System used within the emulation environment is presented.

The reconstruction of the emulation environment using the data collection component is visualized in Figure 3.12, directly showing the results of the computation of the network inventory, the vulnerability inventory and the reachability matrix.

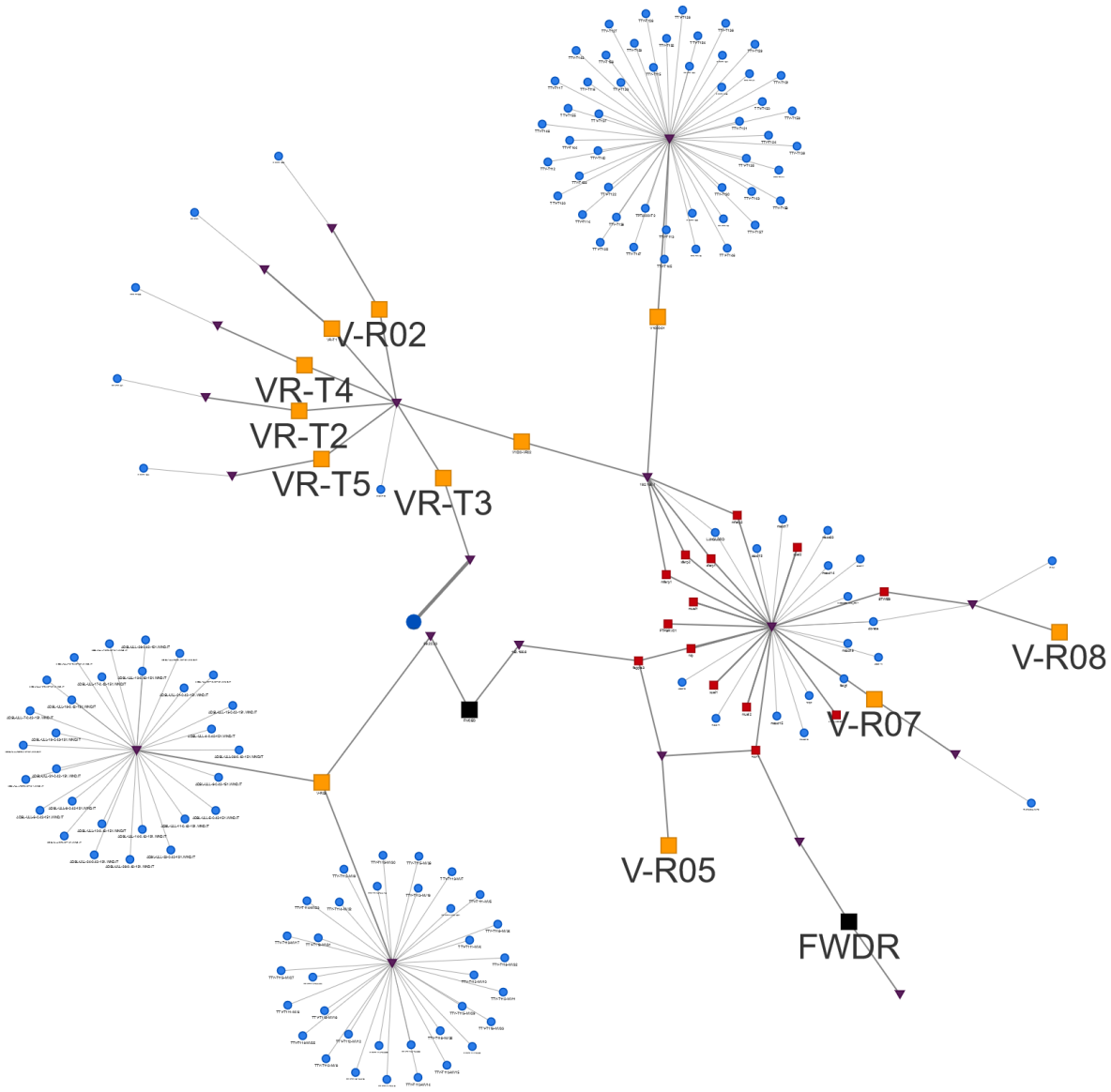


Figure 3.12: Reconstruction of the emulation environment using the data collection component.

The tested version of the emulation environment is composed by 166 nodes, divided into 19 broadcast domains (purple triangles). It is possible to identify a set of routers (yellow squares which connect the broadcast domains). Routers and firewall (black squares) imply firewalling rules on network traffic, in order to protect the command and control network from intrusion.

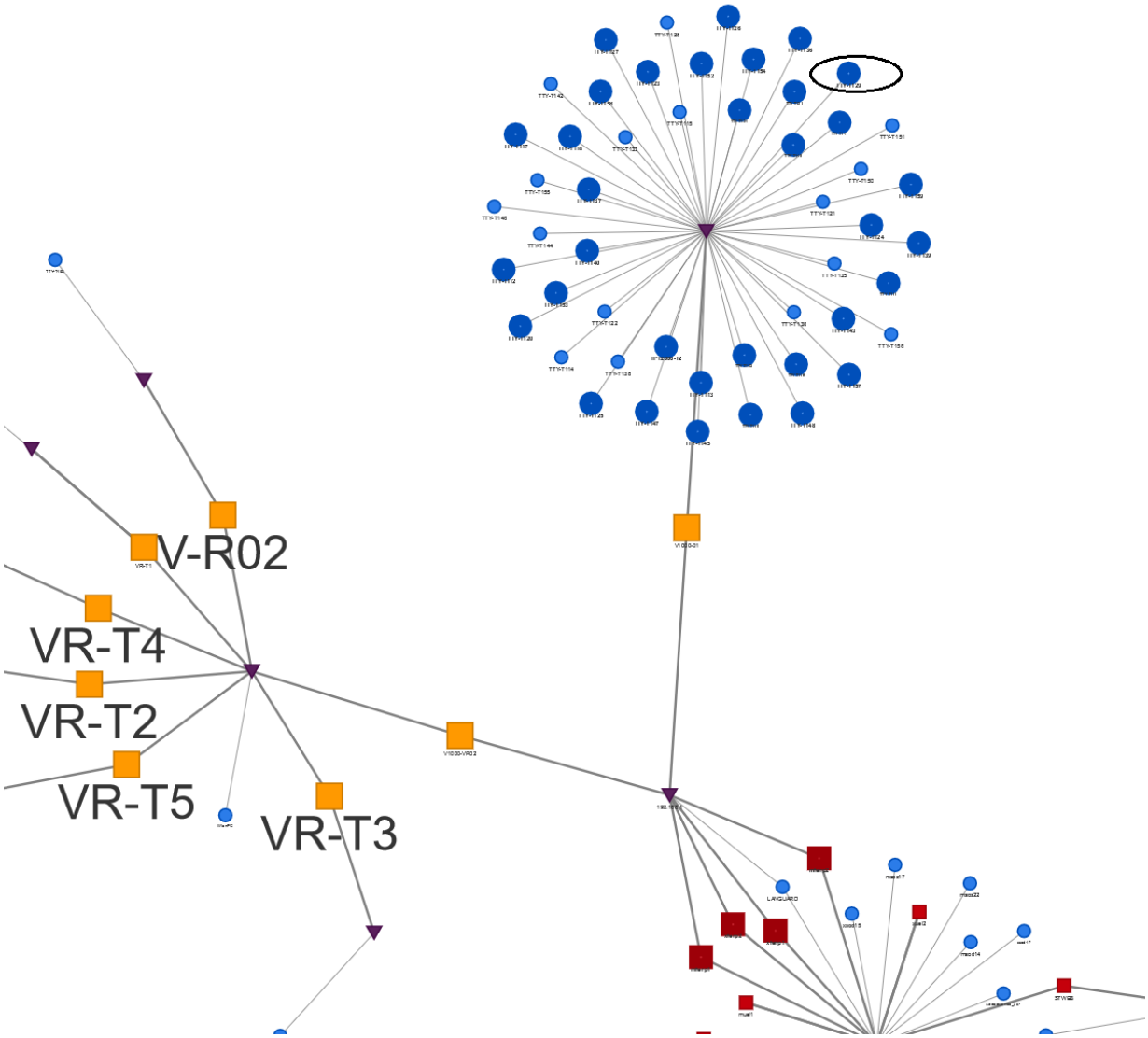


Figure 3.13: Computed reachability for each node starting from the node with the circle.

It is also possible to visualize the computed reachability for each node, as an example in Figure 3.13 indicates. Each node of the monitored system can be selected (in Figure 3.13, we circled a selected node) in order to check all reachable nodes (highlighted nodes). Information shown for generating the diagram in Figure 3.13 is extracted from the computed reachability matrix. As an example, an extraction from the reachability matrix visualized in Figure 3.13 is provided in the listing below.

The highlighted IP address (see the IP in bold) belongs to a node close to the command and control network which is reachable from our source node only through port TCP 21, 22, 111 (since the router in the communication path enforces a set of firewalling rules which blocks any other protocol).


```
"node_Ident" : "d3480ddc-fe4a-4b94-9dc5-5cf94e658291",
        "interface_Ident" : "b7dd065c-1e00-4394-b776-7e13e7c17b1e",
        "address" : {
          "ident" : "510da421-7f1d-4793-8600-7e8caf8fb568",
          "netmask" : "255.255.255.0",
          "address" : "192.168.1.4",
          "category" : "IPV4",
          "vlan_name" : "",
          "vlan_num" : 0
        },
        "responsiblePath" : [ {
          "responsibleNodeList" : {
            "node_Ident" : [ {
              "rank" : 0,
              "node" : "a662254d-50d0-485c-ab9f-5f9c6a6b61d9"
            } ]
          },
          "protocolList" : {
            "protocol" : [ {
              "name" : "Secure Shell (SSH)",
              "version" : ""
            }, {
              "name" : "FTP - control (command)",
              "version" : ""
            }, {
              "name" : "Sun Remote Procedure Call",
              "version" : ""
            } ]
          },
          "portList" : {
            "portRange" : [ {
              "ip_Protocol" : "TCP",
              "port" : "22"
            }, {
              "ip_Protocol" : "TCP",
              "port" : "21"
            }, {
              "ip_Protocol" : "TCP",
```


Reactivity Experiments

The data collection component has been evaluated in terms of (re)computation times after changes in the monitored system. The ACEA data sources (detailed in Section 3.2) are outside of the scope of the PANOPTESEC project. Hence, scan times are not part of this analysis. A detailed analysis over scan times is reported in D7.4.2. All data collection experiments were run on an Ubuntu Virtual Machine with 4 cores (2297.339 MHz) and 16 GBs of RAM.

Experiment 1:

A node in the Monitored System is switched off. The Data Collection must be able to detect the change and trigger a new computation of the network topology reconstruction.

As explained in Section 3.2, the network inventory processor performs a check over the data sources, searching for changes in the monitored system using a configurable timer, which has been set to 8 minutes for this particular test.

When the timer triggers, the network inventory processor is able to detect (using as a data source the Whatsup Gold Network management system) that a known node has been shut down. A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerability mapping enrichment and for the computation of the vulnerability inventory and the scored vulnerability inventory. Once these data are computed (and concurrently persisted in the Persistency Manager), the new network inventory is made availabe to the Reachability Matrix Correlator, which computes the reachability matrix. In this setting, 4 different iterations of this experiment are conducted, as reported in the following table:

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time
Test 1	38s	43s	385s	469s
Test 2	36s	45s	378s	462s
Test 3	36s	42s	381s	461s
Test 4	40s	48s	390s	480s

Start time: when the internal timer of the network inventory processor triggers.

End time: when the reachability matrix is received by the Persistency Manager.

The overall data collection computation time, of course, takes into consideration also the transmission times over the messaging system of the integration framework (the size of the reachability matrix file is 20.1 MB) and the (small) overhead imposed by the integration framework and its Java wrappers around each component.

Experiment 2:

A node is added to the monitored system. The data collection must be able to detect the change and to trigger a new computation of the network topology representation.

As explained in Section 3.2, the network inventory processor performs a check over the data sources, searching for changes in the monitored system using a configurable timer, which has been set to 8 minutes for this particular test as in the previous experiment.

When the timer triggers, the network inventory processor is able to detect from one of its data sources (this depends on which of them is the most up to date) that a known/unknown node has been added to the monitored system. A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerabilities mapping enrichment and for the computation of the vulnerability inventory and the scored vulnerability inventory. Once these data are computed (and concurrently persisted in the Persistency Manager), the network inventory is transmitted to the reachability matrix correlator, which computes the reachability matrix. 4 different iterations of this experiment are conducted, as reported in the following table:

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time
Test 1	43s	47s	385s	477s
Test 2	42s	46s	378s	467s
Test 3	45s	47s	381s	476s
Test 4	42s	47s	390s	481s

Again, the overall Data Collection computation time takes into consideration also the transmitting time over the messaging system of the integration as well as the overhead imposed by the integration framework.

Experiment 3:

A port is closed on a node of the Monitored System. The Data Collection must be able to detect the change and trigger a new computation of the network topology reconstruction.

When the timer triggers, the network inventory processor is able to detect from one of its data sources (in this case it will likely be a fast topology scanner) that a port (in this specific case, the FTP port on a medium voltage substation) has been closed on a device of the monitored system. A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerabilities mapping enrichment and for the computation of the (scored) vulnerability inventory. Once these data are computed (and stored), the network inventory is transmitted to the Reachability Matrix Correlator, which computes a new version of the reachability

matrix. 4 different iterations of this experiment are conducted, as reported in the following table (experimental setting the same as for the previous experiments).

:

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time
Test 1	52s	43s	381s	477s
Test 2	56s	43s	379s	481s
Test 3	54s	46s	385s	487s
Test 4	52s	45s	388s	488s

Experiment 4:

A firewall rules on a router of the Monitored System is changed. The Data Collection must be able to detect the change and trigger a new computation of the network topology reconstruction.

When the timer triggers, the network inventory processor is able to detect (using as a data source the database containing all firewall rules for routers/firewall inside the monitored system) that firewall rule has been changed on a known router (from Figure 3.13, the router connecting the source node with the internal network will now block all communication via port 111 from the source node domain to the inner domain). A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerabilities mapping enrichment and for the computation of the (scored) vulnerability inventory. Once these data are computed (and stored), a new reachability matrix is computed as well. 4 different iterations of this experiment are conducted, as reported in the following table (time values include computation and data transmission times):

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time
Test 1	37s	43s	378s	461s

Test 2	37s	42s	380s	462s
Test 3	39s	46s	378s	464s
Test 4	38s	49s	383s	472s

As a conclusion, it appears that the Data Collection set of components is able to work together in order to reconstruct the network topology of the Monitored System. The reachability matrix usually takes around 6 minutes in order to compute the reachability of the nodes of the emulation environment.

Scalability experiments

The data collection components have been evaluated in terms of analysis of the computation times in case of bigger networks. Data used are a combination of data from the emulation environment and scans of medium voltage substations from the production environment. The topology of the emulation environment is very close to the topology of the considered production environment: These medium voltage substations are connected to the same router also in the production network. All data collection experiments were carried out on an Ubuntu Virtual Machine with 4 cores (2297.339 MHz) and 16 GBs of RAM. The aim of these tests is to show how the data collection set of components scale.

Experiment 1:

In addition to the 166 nodes of the emulation environment, other 335 nodes scanned from the production environment are added to the Medium Voltage set of substations, bringing the total number of nodes at 501.

As explained in Section 3.2, the network inventory processor performs a check over the data sources, searching for changes in the monitored system, with a configurable timer, which has been set to 16 minutes for this particular test.

When the timer triggers, the network inventory processor is able to collect the data from the emulation environment and from the additional (static) scans of the production substations. A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerabilities mapping enrichment and for the computation of the (scored) vulnerability inventory. Once these data are computed and stored, a new reachability matrix is computed (in this experiment 40.3 MBs). 4 different iterations of this experiment are conducted, as reported in the following table (times values include data transmission times):

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time

Test 1	64s	79s	866s	1013s
Test 2	68s	78s	880s	1029s
Test 3	64s	79s	871s	1018s
Test 4	65s	77s	878s	1023s

Figure 3.14 shows a logical representation of this network.

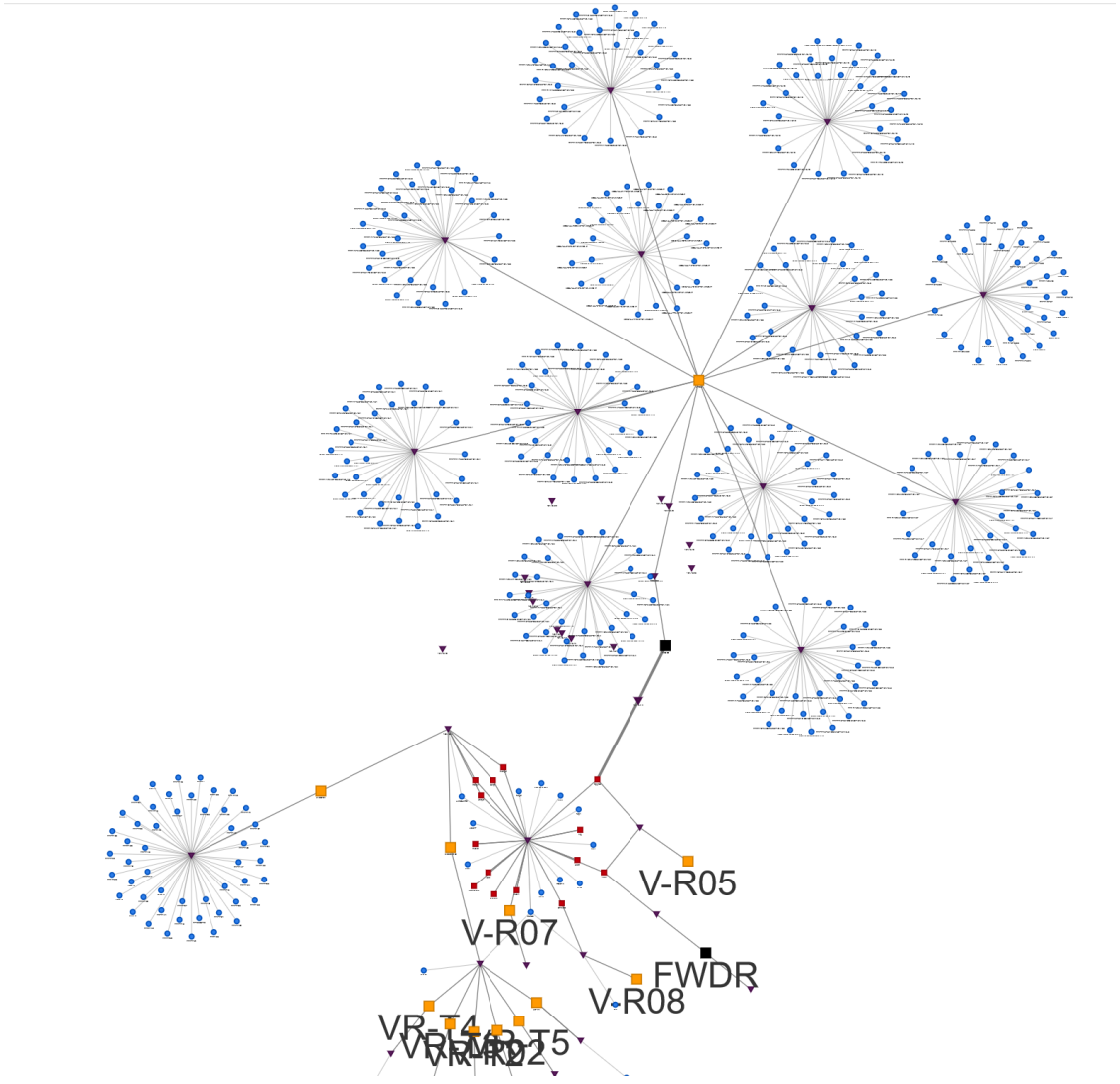


Figure 3.14: Logical representation of the network from scalability experiment 1.

Experiment 2:

In addition to the 166 nodes of the emulation environment, other 1015 nodes scanned from the production environment are added to the Medium Voltage set of substation, bringing the total number of nodes at 1181.

The timer triggering the search for changes in the monitored system is set to 40 minutes. When the timer triggers, the network inventory processor is able to collect the data from the emulation environment and from the additional (static) scans of the production substations. A new network inventory is then computed and transmitted to the vulnerability inventory processor responsible for its vulnerabilities mapping enrichment and for the computation of the (scored) vulnerability inventory. Once these data are computed and stored, a new reachability matrix is computed (87.5 MBs in this experiment). 4 different iterations of this experiment are conducted, as reported in the following table:

	Network inventory computation time	Vulnerability inventory / Scored vulnerability inventory computation time	Reachability matrix computation time	Data collection computation time
Test 1	103s	132s	2033s	2273s
Test 2	101s	130s	2012s	2250s
Test 3	98s	124s	2005s	2230s
Test 4	105s	135s	2056s	2232s

Figure 3.15 shows a logical representation of this network.

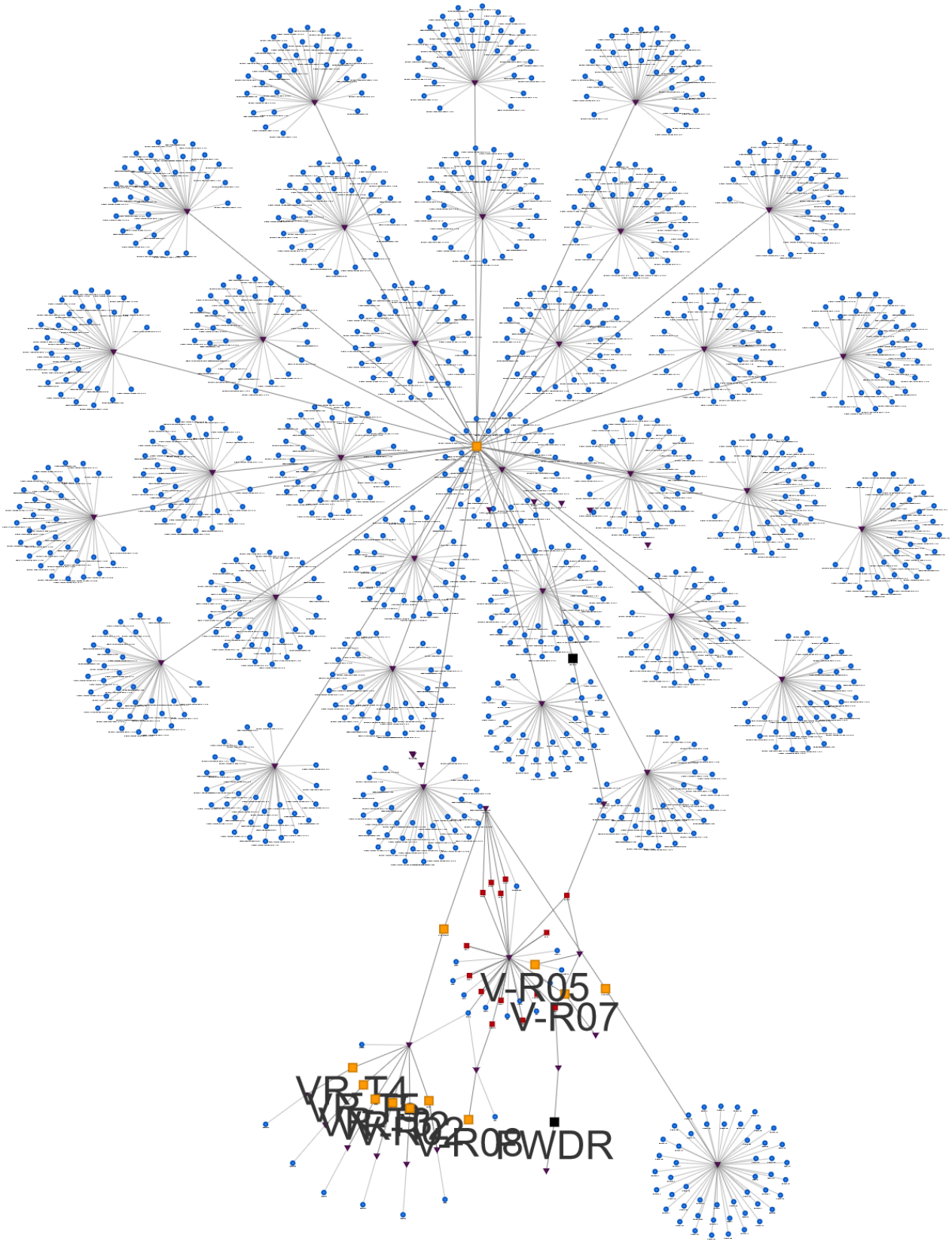


Figure 3.15: Logical representation of network used in the second scalability experiment.

As a conclusion it can be seen that the network inventory processor and the vulnerability inventory processor can easily scale up to around 1200 nodes maintaining good computation timings. The Reachability Matrix Correlator performances with the amount of considered nodes. Overall timings, however, stay under 60 minutes, which is the goal for the PANOPTESEC system. In addition, since this is mainly a computation problem, it is possible to dramatically improve performances by using a dedicated more powerful virtual machine for the Reachability Matrix Correlator. The final integration months for the PANOPTESEC system will be also spent in order to improve the performances of the Reachability Matrix Correlator, with the goal of reducing up to 1/3 the computation time.

4 LOW-LEVEL CORRELATION

4.1 Introduction

The primary purpose of the Low Level Correlation component (LLC) is to normalize events collected from multiple sources into a common data format. In addition, events are enriched by adding information (e.g., vulnerability information) relevant for subsequent processing components in the overall PANOPTESEC processing chain (see Figure 4.1). Here we focus on the integration of LLC with so-called high-level event correlators (actually developed in WP5), which are responsible for identifying attack path instances based on LLC events (and associated vulnerabilities). Normalization, enrichment, and reduction of the overall number of incoming events (from the perspective of high-level online correlation) are the main requirements of LLC.

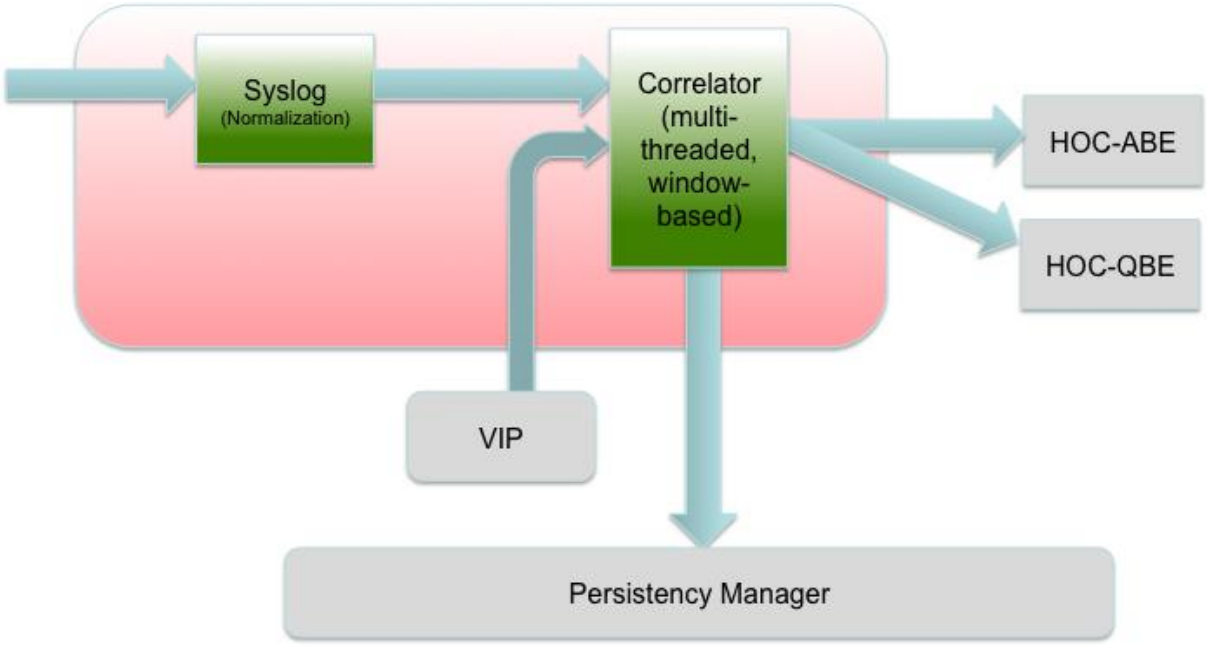


Figure 4.1: Schematic illustration of LLC integrated into the overall PANOPTESEC architecture.

A secondary purpose of LLC is to group alerts such that also suspicious activities are characterized which try to exploit unknown (or not yet reported) vulnerabilities (e.g., zero day attacks). For instance, in order to detect investigative attacks such as port scans, LLC merges events with the same source or destination address (occurring within a certain time

window). Thus, LLC's output serves the purpose of situational awareness and can be displayed to operators or can be stored in the Persistency Manager for forensic purposes.

The LLC component consists of a Syslog engine which normalizes incoming heterogeneous Syslog message streams with the help of a multi-threaded, window-based approach (see also Figure 4.1). Syslog messages contain events sent by probes deployed within the monitored network. In addition, LLC receives vulnerability information from the network inventory (vulnerability information is updated regularly by the vulnerability inventory processor, see the previous section). LLC output is forwarded to both high-level online correlation modules, which are called Automata Based Engine (ABE) and Query Based Engine due to the specific approaches used to implement attack path instantiation functionality. LLC output is referred to as LLC alerts or alerts for short. To allow ABE and QBE to process these alerts, vulnerability information about the threatened application or host needs to be added from the network inventory.

The main techniques implemented in LLC for event prioritization and correlation based on pattern mining techniques are explained and analyzed in detail in Appendix B. An evaluation of the proposed techniques based on data mining is presented in Appendix C. Appendix D contains a description for a system (called Mission Oriented Network Analysis, MONA) supporting automated techniques for deriving a deep understanding of network activities in terms of data structures that help in event correlation. All three appendices are published as conferences papers (see the appendices for citation information). In this section we provide details about alert processing that are relevant for understanding the integration of the LLC prototype into the PANOPTESEC system.

4.2 Event Processing

Probes of multiple distributed firewalls (FWs), intrusion detection systems (IDSs) or intrusion prevention systems (IPSs) are collected throughout a monitored network such that large series of log entries coming inside Syslog messages need to be merged (or fused). As different message streams could use different data formats, events need to be normalized before being further processed. In order to implement fused event processing in this fashion, the goal of low-level correlation is to provide for event normalization, enrichment, verification, and merging.

4.2.1 Event Normalization

In the overall LLC architecture, data from the above-mentioned probes are fed into a SYSLOG engine [1] for real-time data processing. Due to the heterogeneous nature of data formats provided by probes in a distributed environment, it is necessary to implement a normalization process to allow for coherent handling of events in subsequent processing steps. Event normalization for the ACEA electrical power grid's communication network test is realized by a SYSLOG configuration as well as the set of descriptive attributes of normalized alerts is inspired by the Intrusion Detection Message Exchange Format (IDMEF). Events are enriched when values for required attributes are not provided by original message sources such that sensible default values are inserted. After normalization and enrichment, LLC proceeds with event verification. In the following we describe details on verification, using introductory examples with normalized alerts.

4.2.2 Event Verification

Event verification is subdivided into three parts, namely event vulnerability verification, event severity verification, and event operational impact verification. We first discuss vulnerability verification, with the central idea to relate events with information about known vulnerabilities for the network under consideration. Vulnerabilities are available from the network inventory, listing all monitored network devices and detected vulnerabilities. As indicated in Section 3, to detect vulnerabilities NMAP and OpenVAS periodically scan the monitored network. The low-level correlation process relies on this knowledge base to link normalized events to the monitored network. Normalized events can have many fields and in the following we focus on special cases in order to demonstrate central ideas using several examples.

4.2.2.1 Event Vulnerability Verification

Consider, as shown in Table 4.1, an event with a reference to a local exploit reported by an IDS with the analyzer ID CEDET01IDS. Local exploits take advantage of vulnerabilities or bugs. For the example in Table 4.1 we assume that an IDS (CEDET01IDS) has generated an event referring to CVE vulnerability.

Table 4.1: An alert issued by an IDS probe with the analyzer ID CEDET01IDS reports a local exploit.

Analyzer ID	Event ID	Time	Source	Destination	CVE ID
CEDET01IDS	1	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	CVE-2016-0034

An exploit can only be successful if the respective vulnerability or bug is indeed present on the targeted network device. Thus, event vulnerability verification needs to check whether the vulnerability has been detected on the targeted device by extracting respective information from the network inventory. To continue the example, we assume that data shown in Table 4.2 are available, indicating that CVE-2016-00034 is indeed associated with network device 132.8.1.5.

Table 4.2: Vulnerability information extracted from the network inventory.

Network Device	CVE ID
132.8.1.5	CVE-2016-0034

The network inventory allows the event vulnerability verification process to link a reported local exploit, as shown in Table 4.1, to vulnerabilities detected on source or destination hosts as shown in Table 4.2.

Table 4.3: A verified vulnerability alert with an added tag VULNVERIFIED.

Analyzer ID	Alert ID	Time	Source	Destination	CVE ID	Classification ID
CEDET01IDS	1	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	CVE-2016-0034	VULNVERIFIED

If an event can be linked to a previously recorded vulnerability, an LLC alert is generated with the added classification ID VULNVERIFIED. Table 4.3 shows an alert with the added classification ID VULNVERIFIED. In order to reduce the workload of subsequent components, in the LLC configuration there is an option to suppress events for which no vulnerability can be identified. FW/IDS/IPS probes report on abnormal behavior occurring in a monitored network, and not only local exploits potentially with a high impact are being detected. The level of severity that probes assign to a reported events is investigated by event severity verification.

4.2.2.2 Event Severity Verification

Given a sensor is certain of abnormal activity with an impact occurring, an event with a severity data field is issued. Should the LLC be unable to verify an event's vulnerability, but there is an impact associated with the event then an LLC alert is generated with an added classification ID UNVERIFIED. An alert's vulnerability can remain unverified due to two reasons: (i) An event reports an exploit using a particular CVE ID, which for the network inventory cannot provide information, or (ii) the network inventory detects a vulnerability on source or destination, which is not mentioned in the event. Consider, as shown in Table 4.4, for example a situation with an event associated with a CVE ID "CVE-2016-0033" and severity "medium" is reported by an IDS with the analyzer ID CEDET01IDS.

Table 4.4: An event issued by an IDS probe with the analyzer ID CEDET01IDS reports an abnormal activity with a severity and CVE ID.

Analyzer ID	event ID	Time	Source	Destination	CVE ID	Severity
CEDET01IDS	2	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	CVE-2016-0033	medium

CVE-2016-033 is not a vulnerability that was detected on source or destination. Hence, the event does not have a verified vulnerability. However, given that an event contains a severity, an LLC alert is generated for further consideration with a tag "NOTVERIFIED" as shown in Table 4.5.

Table 4.5: A verified severity alert with an added tag NOTVERIFIED.

Analyzer ID	Alert ID	Time	Source	Destination	Severity	Classification ID
CEDET01IDS	2	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	medium	NOTVERIFIED

In comparison consider that an event reports a medium severity incident with no associated vulnerability as described in Table 4.6.

Table 4.6: An event issued by an IDS probe with the analyzer ID CEDET01IDS reports an abnormal activity with a severity.

Analyzer ID	Event ID	Time	Destination	Source	Severity
CEDET01IDS	2	2016-01-24 11:02:31.20	132.8.1.5	85.1.1.8	medium

The network inventory lists vulnerability “CVE-2016-0034” on the destination network device.

Table 4.7: Vulnerability information extracted from the network inventory.

CVE ID	Network Device
CVE-2016-0034	132.8.1.5

Although the vulnerability could not be verified, an LLC alert is generated with medium severity by the severity verification process with an added classification ID “NOTVERIFIED”.

Table 4.8: A verified severity alert with an added tag NOTVERIFIED.

Analyzer ID	Alert ID	Time	Source	Destination	Severity	Classification ID
CEDET01IDS	2	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	medium	NOTVERIFIED

4.2.2.3 Event Fusion

SYSLOG event messages may be indications for abnormal activities in a monitored network. As argued above, some events are clearly linked to an exploited vulnerability and LLC alerts are generated. However, the vast majority of events are simply reports of abnormal activities in a network that may or may not represent alerts.

The purpose of event fusion is to combine events that represent a common attack occurrence. The LLC process maintains a sliding window with some number of messages containing events. The size of the event processing window is computed as the averaged daily number of events occurring every second. This allows for fixed computation time within the LLC process, regardless of how long the component is running. Clearly, it must be taken into consideration that events can only be fused if they are in the same window. The window size can be selected in the LLC configuration file such that scalability and expressivity requirements can be fulfilled in a particular context.

The goal of duplicate event fusion is to combine events representing an identical detection of the same situation by sensors. A single sensor is able to produce duplicate events, when an situation fits multiple rules. This phenomenon is also referred to as event splitting. Table 4.9 describes how an IDS sensor with an analyzer ID CEDET01IDS splits a port scan into two events. The LLC process performs duplicate event fusion by merging them into an aggregated event, collected as an LLC alert. The aggregated LLC alert averages the timestamp of both events and combines all diverging data fields.

Table 4.9: Example for merging duplicate events.

Analyzer ID	Alert/ Event ID	Time	Source	Destination	Description	Classification ID
CEDET01IDS	1	2016-01-24 11:02:31.10	85.1.1.8	132.8.1.5	Port scan	
CEDET01IDS	2	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	Port scan	
CEDET01IDS	{1,2}	2016-01-24 11:02:31.15	85.1.1.8	132.8.1.5		DUBLIMERGE

Taking the same example of port scanning, it is possible that distinct sensors detect the port scan. Given source and destination with an event are identical, the low-level correlation processes performs event fusion. Assume that two distinct IDS sensors with an analyzer id CEDET01IDS and CEDET02IDS both report a port scan. The event correlation process performs duplicate event fusion by merging them into an LLC meta-alert. Again, the aggregated LLC alert averages the timestamp of both events and combines all diverging data fields.

In Table 4.10 there are two port scans shown from one source (1 to n). With an appropriate classification ID a new LLC alert that fuses both port scan events is generated (see the last row in Table 4.10).

Table 4.10: Example for merging duplicate events with the same source address.

Analyzer ID	Alert/ Event ID	Time	Source	Destination	Description	Classification ID
CEDET01IDS	4	2016-01-24 11:02:31.10	85.1.1.8	132.8.1.4	Port scan	
CEDET01IDS	5	2016-01-24 11:02:31.20	85.1.1.8	132.8.1.5	Port scan	
CEDET01IDS	{4,5}	2016-01-24 11:02:31.15	85.1.1.8	{132.8.1.4,132.8.1.5}		SRCMERGE

In Table 4.11 we show a port scan from different source to a single device. The fused LLC alert shown in the last row, again fusion is indicated using a specific classification ID.

Table 4.11: Example for merging duplicate events with the same destination address.

Analyzer ID	Alert/ Event ID	Time	Source	Destination	Description	Classification ID
CEDET01IDS	6	2016-01-24 11:02:31.10	85.1.1.8	132.8.1.4	Port scan	
CEDET01IDS	7	2016-01-24 11:02:31.20	85.1.1.9	132.8.1.4	Port scan	
CEDET01IDS	{6,7}	2016-01-24 11:02:31.15	{85.1.1.8, 85.1.1.9}	132.8.1.4		DSTMERGE

It is possible to configure LLC in such a way that original events are filtered and only the meta-alerts or fused alerts are forwarded as LLC alerts.

4.2.2.4 Alert Operational Impact Verification

Traditionally, the success of security information and event management (SIEM) is determined by the level of protection of critical network devices and applications from attackers. Critical network devices can be routers, servers, etc. and critical applications can compromise database management software or tools for monitoring and controlling an infrastructure. In order to understand how critical an application is for a network's mission, a mission criticality level is added to the network inventory. Per default, application ports that were found responding are listed in the network inventory with a default mission criticality value *Low*. Network operators are able to reclassify an application's mission criticality to *Medium* or *High*. Table 4.12 illustrates an entry from the network inventory.

Table 4.12: An application's mission criticality is added to the network inventory.

CVE ID	Network Device	Application Port	Mission Criticality
CVE-2016-0034	132.8.1.5	80	High

Operational impact analysis identifies all network services and thereby devices, which are potentially affected by a security incident reported in form of an alert. Network operators can predetermine what level of threat to a mission criticality is sufficient for a security incident to be reported.

4.3 Evaluation

To test the vulnerability-centric parts of LLC as well as the source, destination and multiple event merging parts, we focus on the average alert reduction in normal operational mode of the simulation environment. To evaluate the performance of the LLC, metrics with respect to outgoing LLC Alerts $alerts_{LLC}$, incoming Syslog messages msg_{syslog} and Syslog messages that can be attributed to a cyber incident msg_{Syslog}^{atk} are defined. The respective number of occurrences is written as $|alerts_{LLC}|$, $\left|msg_{syslog}\right|$ and $\left|msg_{Syslog}^{LLC}\right|$.

Based on these parameters we define the Syslog message reduction rate as:

$$Syslog\ message\ reduction\ rate = \frac{|msg_{syslog}| - \left|msg_{Syslog}^{LLC}\right|}{|msg_{syslog}|} \times 100 \qquad (1)$$

$$Syslog\ message\ aggregation\ rate = \frac{\left|msg_{Syslog}^{atk}\right| - |alerts_{LLC}|}{\left|msg_{Syslog}^{atk}\right|} \times 100 \quad (2)$$

Given that an attack atk was observed, all incoming Syslog messages that can be attributed to an attack atk, but, however, have not been linked to a vulnerability by the sensor reporting the syslog message, are denoted as $msg_{syslog}^{atkNoCVE}$.

LLC alerts that report $msg_{syslog}^{atkNoCVE}$ are denoted as $msg_{LLC}^{atkNoCVE}$.

Hence, we consider a particular vulnerability has been enriched by LLC, when the incoming Syslog message leading to the LLC alert was not linked to the vulnerability.

$$Vulnerability\ enrichment\ rate = \left(1 - \frac{\left|msg_{syslog}^{atkNoCVE}\right| - \left|msg_{LLC}^{atkNoCVE}\right|}{\left|msg_{syslog}^{atkNoCVE}\right|}\right) \times 100 \quad (3)$$

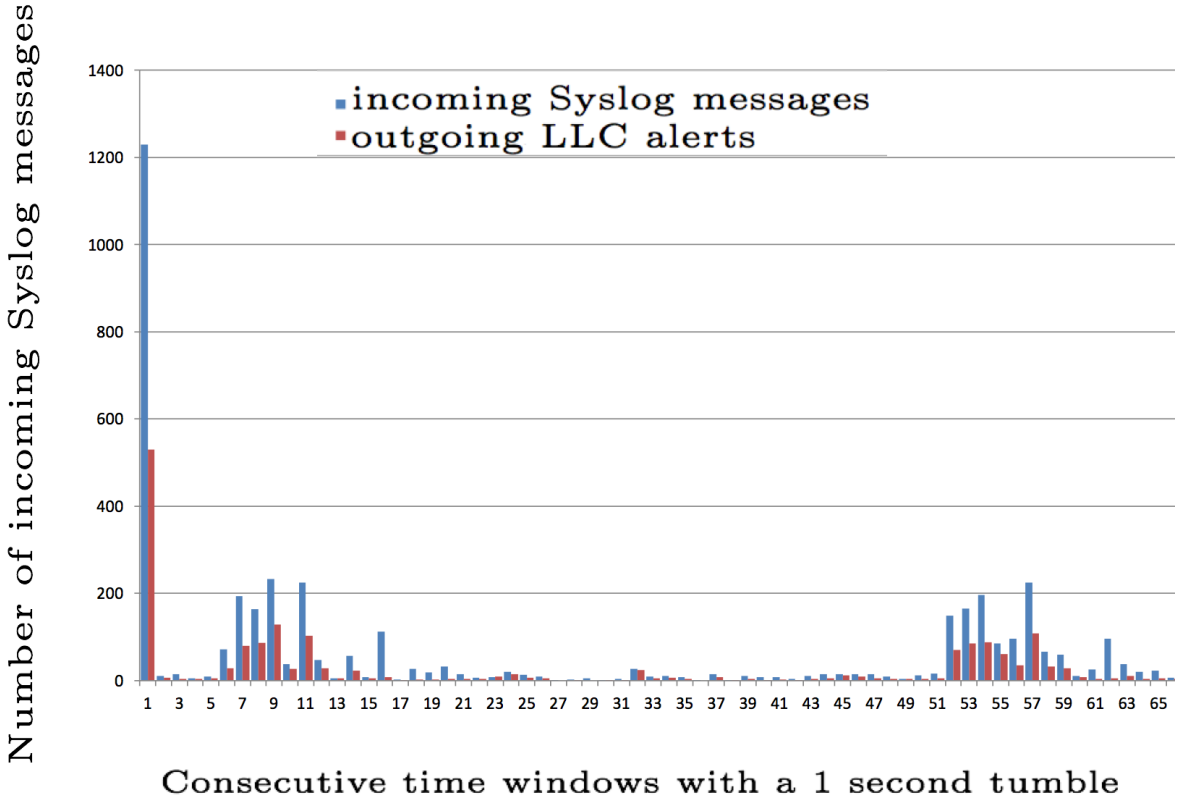


Figure 4.2: Incoming Syslog messages and outgoing LLC alerts in normal, operational mode within the simulation environment.

The currently integrated LLC only focuses on the vulnerability centric evaluation, hence no merged alerts are produced. Additionally, the number of incoming Syslog messages was

reduced from 1500-2000 alerts per second to 5-10 alerts per second. LLC now produces no alerts in normal operational mode.

Figure 4.2 presents the number of incoming Syslog messages and the number of resulting LLC alerts over the time. It can be seen that the outgoing alerts are significantly reduced. In average we achieve an alert reduction rate of 53%. Hence, we satisfy the set goal of 25% regarding the objectives as defined in the Description of Work document for PANOPTESEC.

The Attack Scenario

In order to make these evaluations with tested the LLC+HOC correlation process under a general attack scenario (see Figure 4.3) different experiments were conducted, each one with a specific aim. Moreover, to enforce the test strategy solution different tests on each of these experiments was performed. All tests have a common HOC configuration: QBE has been configured with a threshold **T** of 50%. This determines that when a similarity score of a certain path overcomes **T**, an IAP related to that path is generated. ABE has been configured with a threshold of 60%, and a number of missing alerts of min (1.30%*length of the attack path). The ABE threshold corresponds to the percentage of the length of the attack path we require to begin the emission of IAPs. The number of missing alerts is called hereafter as the parameter k.

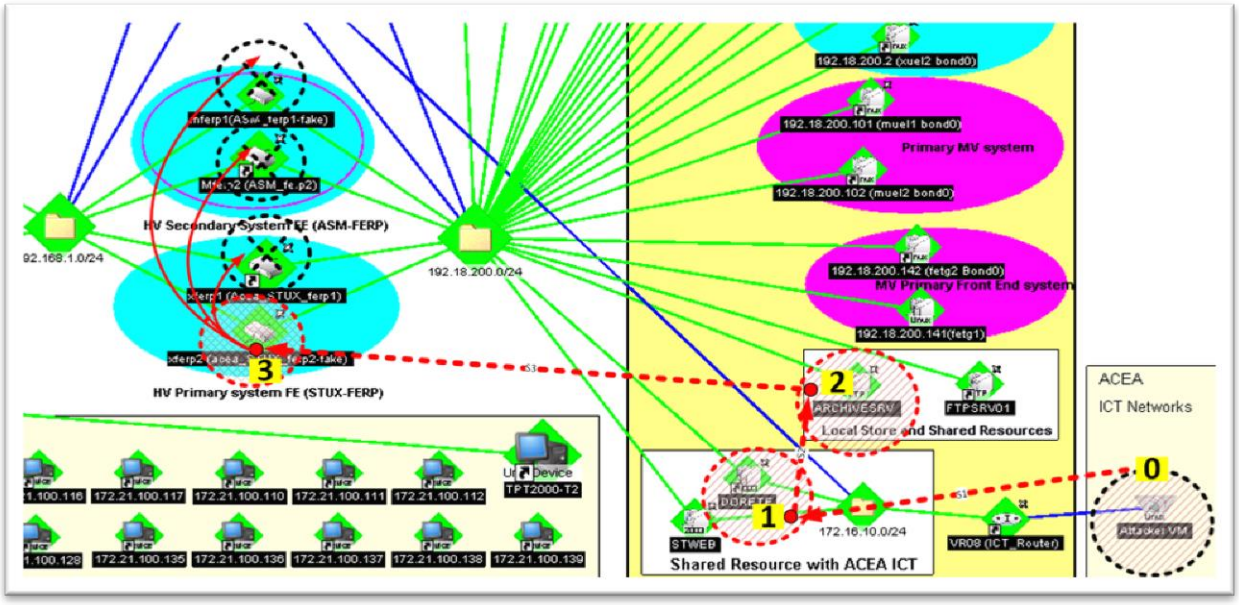


Figure 4.3: Snapshot of the Emulation Environment focused on the nodes involved in the Attack Scenario

The attack scenario shown in Figure 4.3 consists of different attack steps, described in the following:

1. The first step consists of an attack on DORETE server (172.16.10.10). The attacker will use its SMB vulnerability in order to take its control. After that, the Attacker will use DORETE as a gateway for domain 192.18.200.x (pivoting action).

2. Since ARCHIVESRV is the only one able to establish a connection with FERP[1] nodes, the Attacker will obtain ARCHIVESRV (192.18.200.200) control by using DORETE (192.18.200.230) as a gateway.
3. Once ARCHIVESRV (192.18.200.200) is exploited, the Attacker will gain control of FERP (192.18.200.146) through a vulnerability of ftp protocol running on the server. It is mandatory to take into account the fact that this machine can grant access to other FERP nodes.

This attack scenario has been performed on the emulation environment, where the enriched attack graph used, consisted of 20414 different attack paths considering the topology of that day (25th July 2016).
Another important thing that needs to be highlighted is that all the information exchanged among the components are streamed asynchronously over time, therefore they are not computed in different chunks.

Experiment 1:

Reported incidents *SOURCE ->DESTINATION*	Syslog messages corresponding to the potential attack step	LLC Alerts
Step 1: 172.16.10.20 -> 172.16.10.10 (DORETE)	13 3 with vulnerability	1 VULNVERIFIED 1 NOTVERIFIED 2 MULTIMERGE (9)
Step 2: 192.18.200.230 (DORETE) -> 192.18.200.200 (ARCHIVESRV)	1 with vulnerability	1 VULNVERIFIED
Step 3: 192.18.200.200 (ARCHIVESRV)-> 192.18.200.146 (xferp2)	1 with vulnerability	1 VULNVERIFIED
Others:		
192.18.200.146 (xferp2) -> 192.18.200.181 (mferp1)	1 with vulnerability	1 VULNVERIFIED
192.18.200.2 (xuel2) -> 192.18.200.1	5	

[1] Nodes with substring „ferp“ in their names.

(xuel1)		2 MULTIMERGE (4)
TOTAL	4797	9

Overall within experiment 1, 4797 Syslog messages were received and 9 LLC alerts were produced. This results in an overall Syslog message reduction rate of 99.8%. Exxperiment 1 starts at 15:38:25 and lasts until 15:45:49. The previously described attack scenario results in 16 Syslog messages and LLC identifies passes on 14 Syslog messages within 7 LLC alerts. Overall LLC issues 9 LLC alerts during this test and according to Equation 2 results in a Syslog message aggregation of 43.75%. From the 16 Syslog messages, 10 are not linked to vulnerabilities for attack step 1 and LLC enriches them with vulnerability information. This results in a vulnerability enrichment rate of 100%. The enriched vulnerability information is reported within one LLC alert with VULNVERIFIED tag and one LLC alert with the tag MULTIMERGE. In addition, 4 Syslog messages, which cannot directly be attributed as belonging to the previously described attack scenario, are passed on by LLC with 2 LLC alerts. In the following all reported cyber incidents and the corresponding LLC alerts are carefully analyzed.

Within first step of the executed attack scenario, 13 Syslog messages are issued overall. Three of these Syslog messages are linked to vulnerabilities by the sensor reporting the cyber incident; the other Syslog messages are not linked to vulnerabilities. LLC reports 11 of the 13 overall issued Syslog messages. One reported vulnerability could be verified as present on DORETE and results in an LLC alert with the tag verified vulnerability (VULNVERIFIED). Another reported vulnerability cannot be verified and results in an LLC alert with the tag NOTVERIFIED. Within every one-second-time window of LLC, every reported incident between the same source and destination pair is reported once as a verified (VULNVERIFIED) or non-verified (NONVERIFIED) LLC alert. Additional Syslog messages, reporting the same incident are aggregated into LLC alerts with the tag MULTIMERGE. In test 1, this leads to two LLC alerts with the tag MULTIMERGE, which aggregates 7 Syslog messages in total.

Executing the second step results in 1 Syslog message, which is linked to vulnerability CVE-2004-2687 and results in one LLC alert with tag VULNVERIFIED.

The third attack step is linked to vulnerability CVE-2004-2687 and also is reported by LLC as having a verified vulnerability.

An additionally executed attack uses xferp2 as a starting point and targets mferp1. This leads to one Syslog message containing a link to CVE-2004-2687 and this is reported as an LLC alert with the tag VULNVERIFIED.

The executed attack leads to a reported incident between xuel2 and xuel1. A closer analysis of these Syslog message shows reveals tags such as ``Analyzer_Log-Flood-Protection'', ``Analyzer_Compress-SIDs", ``Firewall''. These cyber incidents are all reported by sensor CEDETClusterIPS node 1. These reported incidents result in 2 aggregated LLC alerts with the tag MULTIMERGE.

Experiment 2:

Reported incidents *SOURCE ->DESTINATION*	Syslog messages corresponding to the potential attack step	LLC Alerts
Step 1: 172.16.10.20 -> 172.16.10.10 (DORETE)	1615 816 with vulnerability	1 VULNVERIFIED 39 NOTVERIFIED 33 MULTIMERGE (1086)
Step 2: 192.18.200.230 (DORETE) -> 192.18.200.200 (ARCHIVESRV)	9 1 with vulnerability	2 NOTVERIFIED 1 VULNVERIFIED 2 MULTIMERGE (6)
Step 3: 192.18.200.200 (ARCHIVESRV) -> 192.18.200.146 (xferp2)	8 1 with vulnerability	1 NOTVERIFIED 1 VULNVERIFIED 1 MULTIMERGE (5)
Others:		
192.18.200.146 (xferp2) -> 192.18.200.181 (mferp1)	252 19 with vulnerability	1 VULNVERIFIED 19 NOTVERIFIED 12 MULTIMERGE (243)
192.168.1.4 (mferp2) -> 172.21.170.5, 172.21.170.6	28 28	4 SRCMERGE (8)
192.18.200.2 (xuel2) -> 192.18.200.1 (xuel1)	15	6 MULTIMERGE (14)
192.18.200.200(ARCHIVESRV) -> 192.18.200.230 (DORETE)	2 1 with vulnerability	2 NOTVERIFIED
TOTAL	28477	129

Reducing overall 28477 Syslog messages that were received to 129 LLC Alerts results in a false positive reduction rate of 99.5% according to Eq. 1. The second test starts at 15:55:57 and lasts until 16:28:41. The previously described attack scenario is reported within 1,884 Syslog messages and LLC alerts reports 1396 of these Syslog messages by issuing 94 LLC alerts related to the cyber incident. Overall LLC issues 129 LLC alerts and this according to Equation 2 is a Syslog message aggregation rate of 93.15%. 827 Syslog messages are linked to vulnerabilities and 1057 Syslog messages are not linked to vulnerabilities. The LLC enriches 970 of these Syslog messages with vulnerability information leading to a vulnerability enrichment rate of 91.77%.

Additionally, 12 LLC alerts are issued for 24 Syslog messages. In the following, these results are systematically analyzed by starting with attack scenario step 1.

The first step of the test scenario is reported within 1615 Syslog messages, whereas 816 of the overall 1615 Syslog messages are linked to vulnerabilities. LLC reports 1,126 of these Syslog messages within 73 LLC alerts.

The second step of the test scenario is reported within 9 Syslog messages, where one reported incident is linked to a vulnerability. LLC identifies all these Syslog messages as relevant and passes them on within 5 LLC alerts.

The third step is reported by 8 Syslog messages with one Syslog message being linked to a vulnerability. LLC identifies 7 of these Syslog messages as relevant and reports them within 3 LLC alerts.

The additional attack using xferp2 as a starting point for targeting mferp1 leads to 252 Syslog messages. LLC considers 244 Syslog messages as relevant and reports them within 13 LLC alerts.

Starting at 2016-07-25 15:59:38 until 16:28:38, Syslog message report a cyber incident with source mferp2, which is a front-end SCADA server and destination 172.21.170.5. One second after this cyber incident, a cyber incident with source mferp2 and destination 172.21.170.6 is reported. To our knowledge, these cyber incidents are not related to the executed attack scenario.

As already explained within test 1, we also observe cyber incidents between xuel2 and xuel1. The 15 reported cyber incidents between xuel2 and xuel1 have the same message content as in test 1 and result in 5 LLC Alerts containing 14 merged Syslog messages.

Two cyber incidents report 192.18.200.200 (ARCHIVESRV) targeting 192.18.200.230 (DORETE). One of the Syslog messages contains a link to a vulnerability, which LLC cannot verify as present within the emulation environment. Hence, this cyber incident is passed on through two LLC alerts with the tag NOTVERIFIED.

4.3.1 Network Service Dependencies within the Emulation Environment

The disaster recovery site of the energy distribution network provided by ACEA was available for network traffic analysis. Observed communication between network services within this environment can be seen in Figure 4.4 (see the next page).

As this network emulates a real-life network, true knowledge of all existing and non existing network dependencies can only be assumed. Data communication networks are dependent on third party software and operators do not have complete knowledge. However, ACEA personnel classified all identified indirect network service dependencies, which are shown in Figure 4.5, as true positives.

The node named mferp2 is a communication server for multiple substations, which are identified as TTY-T[116-158]. Also, our evaluation shows that mferp2 is one physical device, but is assigned two IP addresses from two different subnetworks. The communication server mferp2 hosts a network service. Port numbers that identify the network service are appended to the host names. This network service belongs to an application, which sends requests to all substations in order to be updated with current measurement information. Therefore, a network service dependency joins these substations to the communication server. The host mferp2 communicates via muel2 with human-machine interface components (HMIs) msoz19 and mso22. Another HMI msoz17 accesses information about the substations TTY-T[116-158]. For this, first muel1 is contacted, who passes the request on to muel2.

As energy distribution networks are critical infrastructure it needs to be ensured that all communication pathways are always available. For instance, regular back-up servers and associated communication pathways and service accesses were analyzed in order to find communication patterns (direct dependencies, indirect dependencies, see Appendix B, C, D). All identified network service dependencies were verified by ACEA operators as true positives (see also Appendix A, B, or C).

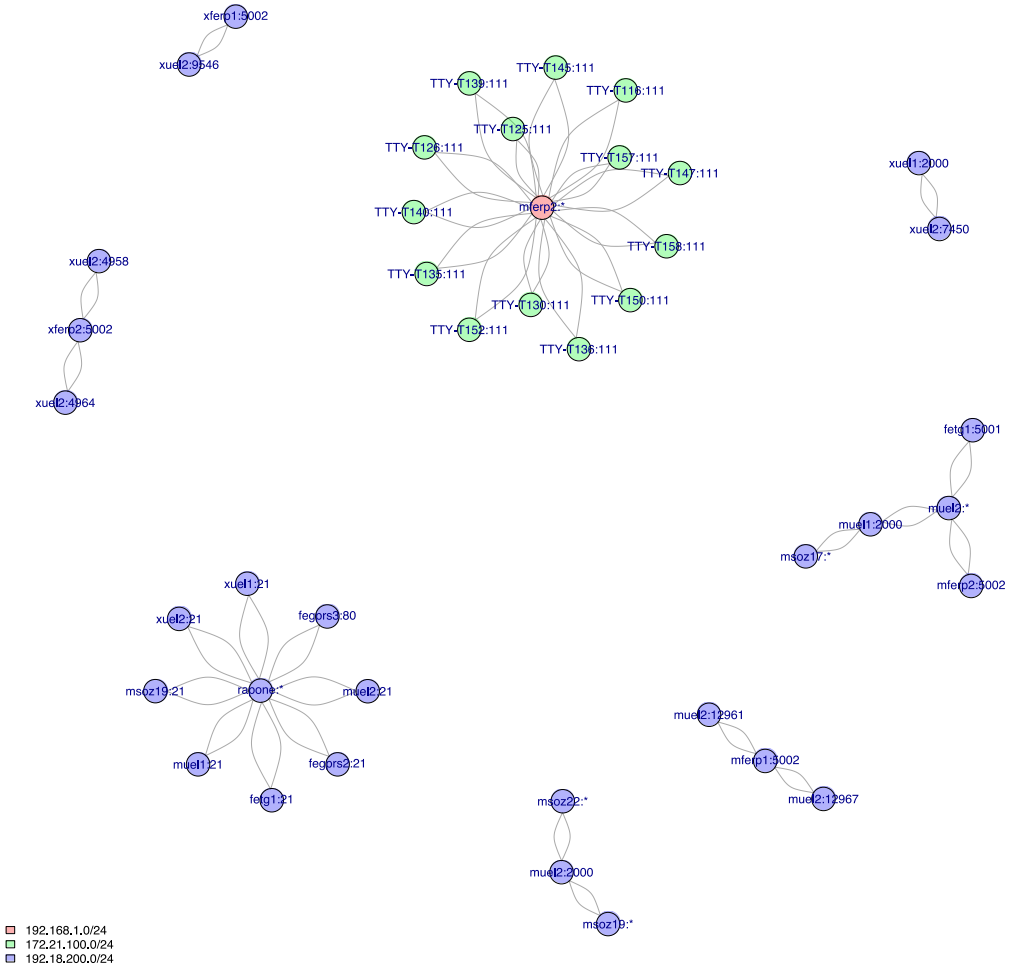


Figure 4.3: Direct network service dependencies (edges) observed within the simulation environment.

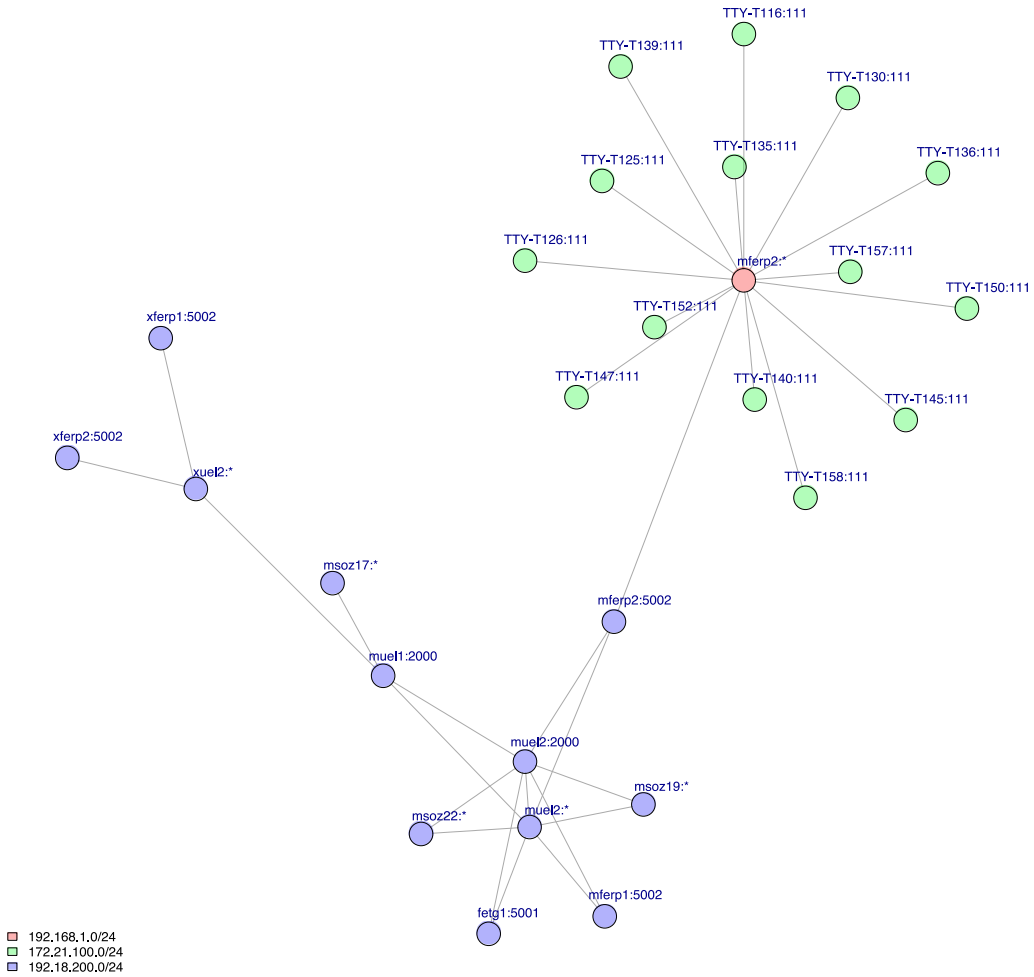


Figure 4.5: Indirect network service dependencies (edges) automatically detected within the simulation environment.

4.3.2 Performance Evaluation of Alert Processing Time

As a number of low-level alerts are being sent from sensors at a high pace, a fairly sophisticated knowledge of both networking technology and infiltration techniques is required to understand them. However, human operators are far too slow for dealing with the numbers of alerts of a real-life system such as the ACEA emulation environment directly.

IDS alert correlation systems try to solve this problem by post-processing the event stream from one or many IDS sensors. The goal of post-processing is to aggregate low-level events and enrich them with vulnerability information as explained above. Assuming a time-based system, the processing time in seconds depends on the number of simultaneously processed

alerts. If the alert processing time exceeds one second, the LLC would not be able to hold that pace, should it continuously occur over multiple time windows. The results of scalability experiments are illustrated in Figure 4.6, which illustrates the processing time in seconds with respect to the number of alerts (Syslog messages) that are simultaneously processed. To demonstrate the parallelizability of LLC, the LLC was deployed in a virtual machine with 4 virtual CPUs and 6 CPUs. The results show a linear increase of computation time for an increasing number of alerts (Syslog messages). A continuous Syslog message rate can be held by LLC if the alert processing time is below one second.

Within the emulation environment, the LLC was tested with a continuous average Syslog message rate of 1500-2000 Syslog messages per second. LLC is able to hold that Syslog rate easily. Hence, this rate is illustrated as a green line to Figure 4.6. The hypothetical upper limit of the production environment is 200,000 Syslog messages per second. This line was added as a green line to Figure 4.6.

Given that the theoretical upper limit of the production environment corresponds to the average continuous Syslog message rate, the task of holding this continuous rate is high risk for the LLC. Much of this risk for classifying 200,000 Syslog messages per second, can be attributed to the Syslog server technology, which is the state of the Art for log management solutions.

Further analysis of the continuous Syslog message rate within the production environment is an ongoing task.

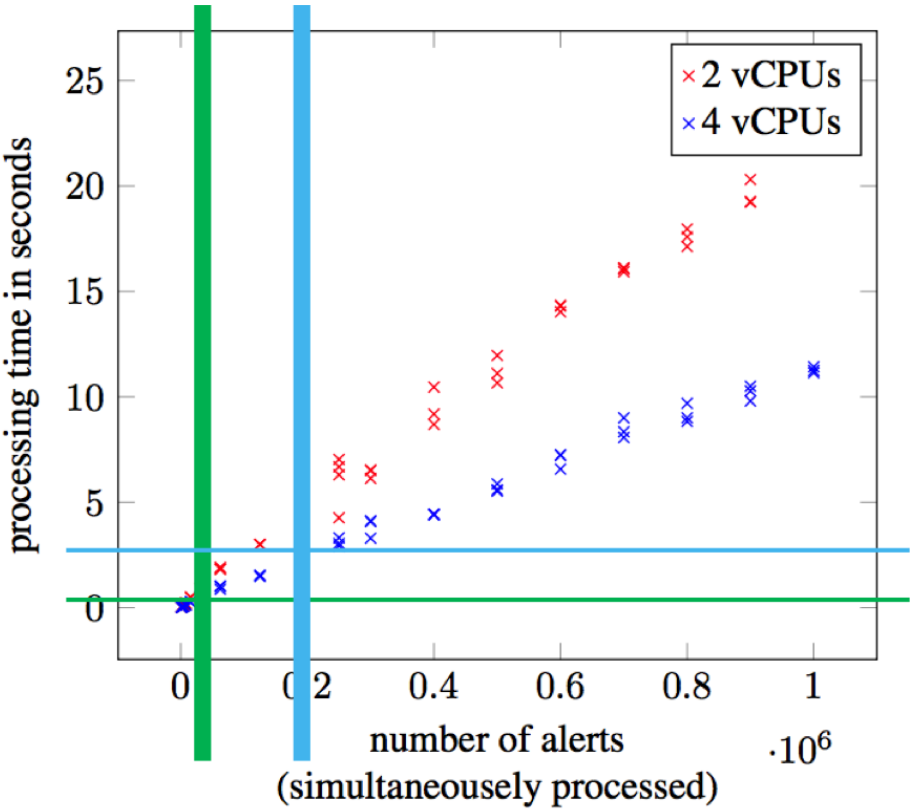


Figure 4.6: Alert processing time.

As has been noted before, the LLC was able to easily handle 2,000 syslog messages per second that occurred during the integration of LLC. Further performance testing in conjunction with the recently delivered ABE and QBE is underway. To test the limitations of LLC as a single component, additional tests with a stand-alone version of LLC were conducted. For this Syslog messages from the emulation environment were replayed continuously at varying alerts per second rates. Based on this, for the LLC as a standalone component, processing up to 10,000 alerts per second on average can be confidently achieved. However, deploying the LLC in an environment with 100,000 alerts per second, or more, is considered high risk.

4.3.3 Performance Evaluation of Network Dependency Analysis

The network dependency analysis is conducted offline. Figure 4.7 evaluates the performance of the network dependency analyzer in a network containing 100 devices and 50 communicating network services. Computing network dependencies requires generating potential indirect dependencies and analyzing whether these indirect dependencies actually exist. To assess, whether more network devices affect the performance, we conducted the same experiment with a network containing 900 network devices. The results of this evaluation are shown in Figure 4.8, revealing the number of network devices does not affect the performance of network dependency analysis.

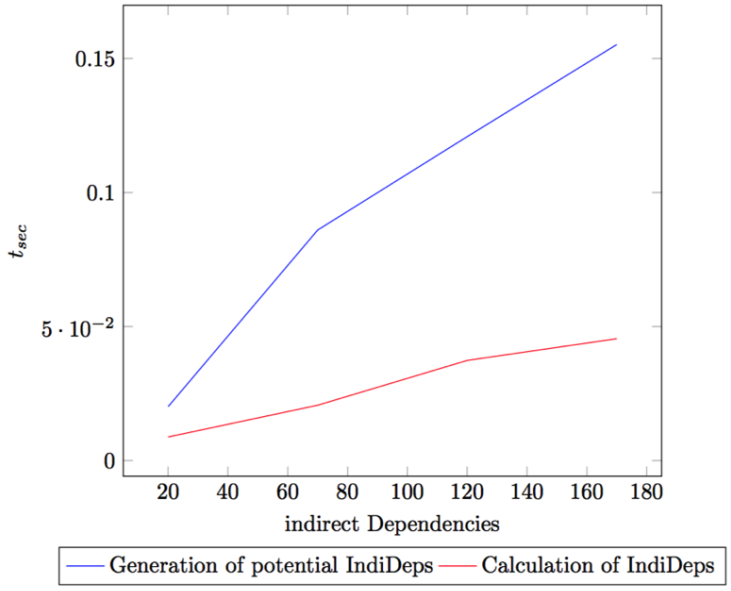


Figure 4.7: Network dependency analysis in a network containing 100 network devices.

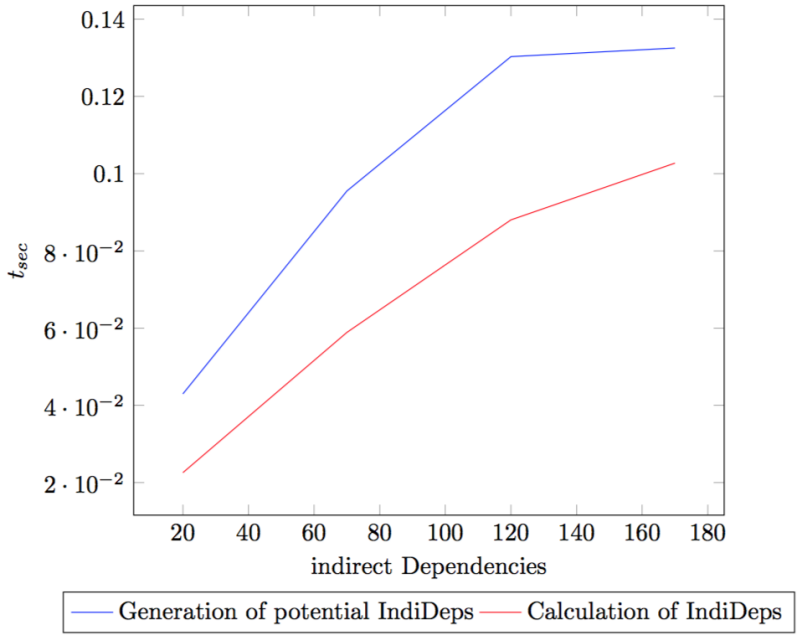


Figure 4.8: Network dependency analysis in a network containing 900 network devices.

4.4 Discussion

We have demonstrated a novel approach to security event management based on a deeper understanding of network activities. This deeper understanding was derived based on network dependency analysis. Network dependency analysis enables deriving a deeper

understanding of network activities and leverages well data-communication networks with different numbers of network devices and indirect dependencies. The proposed frame is able to link network dependencies with security events and thereby assesses a security event's impact on ongoing network activities.

LLC was integrated after a four-week testing period. We come to the conclusion that it was successfully due to a detailed code review conducted by project partners. In addition, ABE and QBE can be driven by the produced LLC alerts.

An additional proof for LLC's successful integration is the successful reactive chain experiment of Attack Scenario 1 discussed above, which was tested in ACEA's premised in Rome. LLC passed on to QBE and ABE relevant evidence for all attack steps used in the scenario.

5 MISSION IMPACT MODELING AND MISSION IMPACT ASSESSMENT

Understanding and assessing potential impacts relevant for a company or mission is a pertinacious problem and a novel research area. For example, a local impact caused by an event on a distant node, say, a user workstation, might lead to a causal chain of operational failures, leading to an impact on a company, as some critical devices were affected required in critical business processes. Such assessments are frequently called *mission impact assessments*.

It is the purpose of the mission impact module (MIM) to perform such mission impacts assessments. To do so, the MIM is based on a probabilistic graphical model that provides a context- and bias-free probabilistic mission impact assessment. It requires a mission dependency model and a shallow network dependency model (SNDM) automatically derived from network traffic analyses by the shallow network dependency analyzer (SNDA) in combination with a network inventory obtained from multiple network scans by the network inventory processor (NIP). A shallow network dependency model represents dependencies between individual resources in a network, e.g., represents dependencies of a webserver on a database server and dependencies of a database master on database slaves. We call such a model "shallow", as only direct dependencies are represented and indirect influences for arbitrary depths are assessed automatically through probabilistic inference but are not made explicit as objects.

Mathematical foundations of the MIM and SNDM and probabilistic mission impact assessment have been presented at the NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact held in Istanbul, Turkey during June 2015 and at the Workshop on Cyber Defence and Security in Bruxelles, Belgium in September 2015. The mathematical foundations and scientific contributions have been published in the NATO unclassified proceedings of the NATO IST-128 Workshop. We deepen and broaden the applicability of probabilistic mission impact assessment in a journal article, currently under review, entitled *"Context- and Bias-Free Probabilistic Mission Impact Assessment"*, which is given as Appendix E. In this article, we present two use cases for MIM and SNDA, namely for response operational impact assessment and for static vulnerability assessments. Furthermore, we present results of the integrated components of MIM and SNDA applied to two real-world usecases.

The first use case is set in the SMIA 2011 environment; an environment set up by Theodor Sommestad at the information warfare lab of the Swedish defense research agency. In this environment a complete network consisting of multiple domains was set up containing multiple ICT servers, clients, mailservers, firewalls, ftps, webservers, even SCADA server, etc. User behavior was simulated inside each domain by action scripts, e.g., checking webservices, emails and downloading files. Intrusion detection systems provided information to one team in charge of monitoring the complete network and noting suspicious behavior. Another team was in charge of infiltrating the network. Both teams carefully documented their approaches, all network traffic was recorded over six days, and the infrastructure was well documented, which all provides an excellent ground truth for our assessments. In this environment we apply a

static vulnerability assessment and automatic network dependency analyses. In Appendix E we show that the automatic analysis delivers highly satisfying results, that parameters are directly validatable and interpretable, and that obtained impact assessments are meaningful and raise one's situational awareness significantly for the targets that were compromised by the attacking team. In the SMIA 2011 use case, the integrated interaction of MIM, SNDA, NIP and vulnerability inventory processor (VIP) is demonstrated locally. Notwithstanding, we deeply evaluate an employed approximation algorithm used for approximate probabilistic inference in Section E.4, and verify the linear scalability of the MIM on extremely largely scaled domains based on numerous randomly generated test sets.

Moreover, in Appendix E we present results of the integrated MIM, SNDA and NIP components in the ACEA use case coming directly from the integration framework running at ACEA. The obtained results can be summarized as follows: Business experts are able to design mission dependency models directly, and business experts do not need to be trained, e.g., in probabilistic inference or modeling, to do so. A required mission dependency model was directly created by business experts from the ACEA use case from a production perspective in less than three hours. Results show that a model designed for a production environment is applicable to the emulation environment. In particular, a mission dependency model encompasses all related information about a company, their missions and processes, and critical resources threatened by potential events. The requirements for these information have been developed from state-of-the-art models, as described in Section E.6, and in deep collaboration with work package 5 (WP5) partners in charge of the attack graph generator (AGG). Moreover, results obtained from the SNDA, i.e., a shallow network dependency model, have been assessed to reasonably represent the ACEA production environment by IT security consultants to the ACEA company. Moreover, the automatically learned dependency model was validated to bear reasonable dependencies. The visualized results of the ACEA use case in Appendix E (Figure E.13, for the reader's convenience repeated here in Figure 5.1) are directly extracted from the PANOPTESEC environment running inside ACEA Distribuzione and show the deep integration of PM, MIM, SNDA and NIP. To emphasize this, the created and learned models, i.e., mission dependency model and shallow network dependency model, were obtained automatically from and used in a running integration of the PANOPTESEC framework inside the emulation environment. These modules were validated by internal and external experts of the ACEA use case, who validated both models from a view on ACEA Distribuzione, i.e., by a view from a production environment. From that perspective, results validate that the emulation environment represents the production environment and that the integration is working as expected.

The integrated MIM and SNDA components provide submodules for response operational impact assessment (ROIA), targeted towards assessing potential unintentional damage caused by potentially executed response plans. Inside the PANOPTESEC system, the strategic response decider (SRD, described in deliverables of WP5, e.g., D5.4.2), proposes response plans from a technical and financial perspective, minimizing an attack surface while being a financially attractive investment. To do so the integrated SRD component utilizes a response financial impact assessment (RFIA) based on return on response investments (RORI) values. An RFIA does not need to be necessarily inline with an ROIA and is executed from a contrary perspective

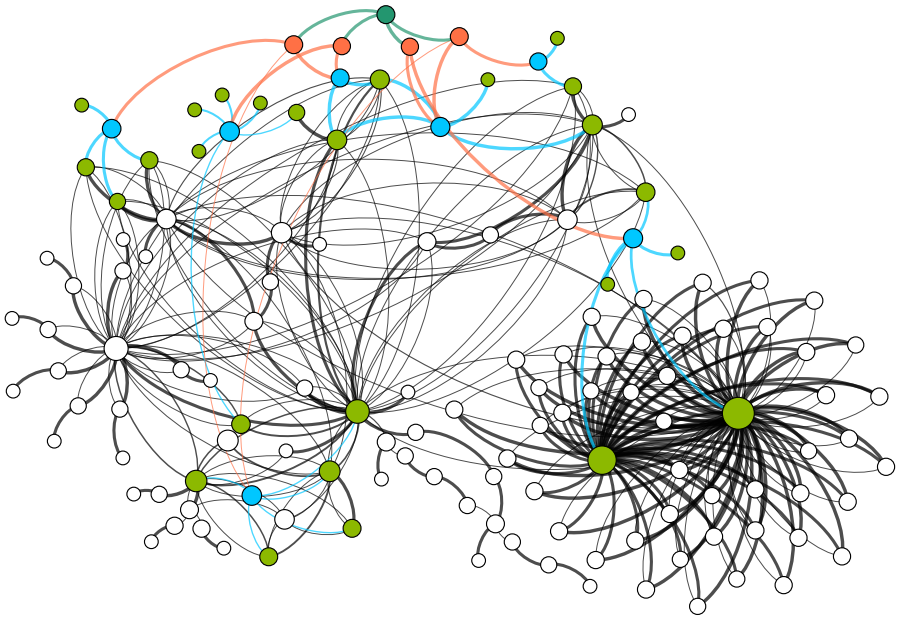


Figure 5.1: Resource dependency model extracted from one day traffic captures of the emulation environment within ACEA Distribuzione (dark green), where related critical devices are highlighted in green, business functions in blue, business processes in orange. This model was validated and verified to be reasonable by the company's IT experts. $n_N = 344$**,** $n_E = 754$**.**

with a contrary intention. In collaboration with WP5 partners a deep requirement for a multi-dimensional assessment of response plans has been identified. As part of the PANOPTESEC project we use both of these impact assessments to choose optimal response plans based on an unweighted best compromise in all dimensions. Approaches towards such an optimization have been established in collaboration with all WP5 partners, pondering multiple approaches, from which an unweighted best compromise search has been chosen in agreement. In the paper *"Selection of Mitigation Actions Based on Financial and Operational Impact Assessments"* (Appendix F, to be published by ARES 2015 at the beginning of September) the interaction of all integrated components of MIM, SNDA, NIP, ROIA and RFIA is demonstrated and applied to the ACEA use case. We discuss and demonstrate obtained results for optimal response plans in a realistic attack scenario and discuss the merits of multidimensional impact assessments in favor of a unidimensional assessment.

The obtained results and insights from Appendix F can be summarized as follows: A response plan evaluation and proposal from only one perspective, i.e., only from an operational or only from a financial perspective does not deliver satisfying results, as expected. Considering only

one assessment solution, too simple or too invasive response plans causing collateral damage are chosen. Multiple evaluations of response plans, taken live from the integration framework, show that a combination of both assessments, i.e., ROIA and RFIA, deliver non-trivial response plans after searching a best compromise from all assessments, as demonstrated in Section F.5.

As outlined in the article *"Context- and Bias-Free Probabilistic Mission Impact Assessment"* (Appendix E), a mission dependency model required in a MIM is directly designable by experts without requiring deep training of probabilistic preliminaries, and also extractable automatically from given BPMN sources, as envisioned for future work on the PANOPTESEC project as part of the business mission collector (BMC) component. When given information from multiple sources, e.g., multiple BPMN models and information gathered from different experts from different expertises, a common problem is eminent: Experts frequently use different nomenclatures, descriptions, languages and references for common entities, inevitably leading to semantic overlaps between information sources due to semantic and terminology gaps. Therefore, a naive concatenation of extracted and gathered business dependency model inevitably leads to duplicate entities and incorrect impact assessments. To obtain well-defined results, i.e., to obtain a solid and consistent business dependency model from multiple sources and experts, a semantic normalization and merging is required for business dependency model. We deeply discuss the semantic normalization and merging problem of mission dependency models in the paper *"Semantic Normalization and Merging of Business Dependency Models"* (Appendix G, to be published by the end of August at CBI 2016.)

Results and insights obtained from Appendix G can be summarized as follows: Information encapsulated inside mission dependency models are required by various components inside the PANOPTESEC framework, such as the MIM and the attack graph generator. In various collaborative meetings between WP4 and WP5 required information was collected and found to be evident from BPMN models. Further, a need to manually create these models from causal perspectives, without requiring deep training of experts and without forcing experts to operate outside their expertise was identified.

For future work on the PANOPTESEC project and follow-up projects, it is envisioned to extend a mission impact assessment towards dynamic domains considering evolutions of impacts over time, predictions into the future and forensic analyzes of impacts in retrospect. The article "Context- and Bias-Free Probabilistic Mission Impact Assessment" (Appendix E) also discusses an extension of the introduced and integrated MIM and SNDA components towards a dynamic mission impact assessment (DMIA). In order to extend a mission impact assessment towards time-varying and time-dependent domains, fundamentally new research on dynamic probabilistic graphical models (DPGM), namely dynamic Bayesian networks (DBNs), had to be undertaken. We identify that classical DBN formalisms are significantly limited in representing indirect causes in certain domains. We introduce a novel form of DPGMs, called Activator DBNs, in the paper *"Indirect Causes in Dynamic Bayesian Networks Revisited"* (Appendix H, published at IJCAI 2015.)

The contribution of Appendix H can be summarized as follows: By introducing activator random variables, we propose template fragments for modeling DBNs under an unrestricted use of time,

anticipating indirect influences on a solid mathematical basis, obeying the laws of Bayesian networks. This is beneficial for application contexts where causal models arise naturally and require a view over time, e.g., automatic learning of causal influences from coarse observation sets and—as a long-term goal—finding causally correct explanations and relations in (temporally uncertain) knowledge bases requiring anticipation of causal chains. For consecutive projects of the PANOPTESEC project, these dynamic probabilistic graphical models build the foundation for a well-defined dynamic probabilistic graphical model for dynamic mission impact assessment, which is predestined for realtime and forensic analysis of mission impacts.

6 CONCLUSIONS

6.1 Significant results achieved

With the main components discussed in this deliverable (NIP, VIP, PM, MIM, RMC, and LLC) we were able to show that one can use a generic architecture (as part of the PANOPTESEC system) to successfully collect data for high-level online correlation (instantiation of attack paths) as well as for analyzing suspicious activities in general. This was achieved by providing an architecture that regularly update context information (from ACEA) in order to drive online data collection and correlation (LLC). Scalability of the integration data collection correlation system was shown using experiments with the so-called emulation system based on the ACEA contingency planning and emergency handling cyber-physical system.

In addition to data collection and correlation, mission impact modeling allows for alert fusion (and this, characterization), global vulnerability score computation (shown with the visualization component) as well as response plan ranking with respect to mission modeling. We put high emphasis on reducing manual effort for setting up the software systems by supporting automatic data interpretation (based on context-specific plugins and carried out regularly) and machine learning techniques to build models used in modules described in this deliverable.

Based on results of WP4, further correlation systems (see WP5 of PANOPTESEC) as well well as the visualization system(s) (WP6) were able successfully carry out their jobs. Verification, validation and integration tests of the WP4 components delivered highly satisfying results for the ACEA use case partner and provided valuable, well-accepted scientific contributions accepted at major conferences, such as IJCAI 2015, ARES 2016, AI 2015, and KI 2016.

6.2 Recommendations

Further scalability experiments can be carried out in the ACEA production environment.

6.3 Deliverable validation

Scientific results have been published at major conferences and workshops in IT security and artificial intelligence. Demos for PANOPTESEC system operations have been give at international exhibitions and at local companies working in the IT security business.

Appendices

A A SEMANTIC APPROACH TO REACHABILITY MATRIX COMPUTATION

Notes *This chapter is a shortened version of a paper that has been published in: Tenth Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS 2015). Ceur Workshop Proceedings, vol. 1523, p. 91-94, ISSN: 1613-0073, Fairfax, VA, USA, 19/11/2015*

A.1 Introduction

This correlation component performs a reachability computation across a monitored ICT network to deduct if two nodes are reachable from each other in the network, for all pairs of nodes representing ICT devices. The Reachability Matrix Correlator (RMC) provides reachability information in a network in the form of a reachability matrix, useful for the Attack Graph Generator component of the Dynamic Risk Management Response System. Several tasks are required to achieve the Reachability Matrix Correlator goal:

1. Produce an abstract machine-readable representation of the knowledge.
2. Study and produce a correct representation of the problem, using the most suitable available methods.
3. Perform applied research to determine the optimum methods to solve automatically the problem.
4. Guarantee that the representation of the knowledge, the representation of the problem and the solving method are abstract enough to be independent of any specific commercial networking devices or applications.
5. Integrate the available tools and methods into a running prototype correlator that produces the correct Reachability Matrix.
6. Guarantee that the IT services provided by the Correlator are be aligned to the needs of the Attack Graph Generator component of the Dynamic Risk Management Response System.
7. Guarantee the compatibility of the technical solution with the overall PANOPTESEC system.
8. Guarantee that the technical implementation of the solution within PANOPTESEC system is performing at an acceptable service level for a prototype within the project scope and quality requirements.

In particular, for task (1), the following characteristics need to be fulfilled:

- Capable of representing all kind of ICT devices, including terminal machines,

- Able to represent connectivity including sub-netting,
- Able to describe ICT devices grouping, according to filtering rules in machines,
- Capable of representing gateways and firewalling rules,
- Having the capacity to represent connectivity between every kind of nodes in the ICT network,
- Able to represent Deployed Access Control Policy Rules (firewall rules, routing rules, NAT rules),

The Reachability Matrix Correlator (RMC) provides algorithms for computing reachability information (aka reachability matrix), required for the Attack Graph Generation module of the Dynamic Risk Management Response System. Reachability information is used to determine if a node can reach another node (via ISO/OSI layer protocols). Some data can be directly retrieved from ICT sources, for computing reachability information algorithms need to be run. Therefore the following requirements have been identified:

- *Determine if a node is reachable from another node on a logical level*. To provide at a logical level if a node can be reached from another node. If in a network a node is reachable from another node, there is a possibility that an adversary might be able to infiltrate a network further. Such information is gatherable from, e.g., Firewall Rules, Mapping Rules, Firewall Logs and/or Traffic Captures.
- *Determine reachability in terms of Source-Port, Target-Port, Protocol*. To obtain a detailed view of reachability in a network and provide the most available information If a node is reachable over a specific port and protocol, there might exist vulnerabilities in such a protocol. Further a node might be reachable, but this reachability does not allow an adversary to progress further in a network.
- *Identify physical entities responsible for a reachability*. Identify hardware entities, e.g. Firewalls, Switches, Routers, that route a reachability on a physical level. A logically non-existing hardware, e.g. a switch, but itself be prone to vulnerabilities, which might allow an adversary to broaden a reachability.
- *Consider that a node might be known via multiple addresses*. To identify a reachability on a logical level between unique devices In a subnetwork, entities might be addressed (e.g. IP) in another way, than from outside the subnetwork.

These requirements are secondary to (a) identifying and defining an adequate representation of knowledge (ontology), the proper Knowledge Base and the appropriate Knowledge Repository [1] to store the Knowledge Base, and (b) defining a proper mode to populate the Knowledge Base.

A.2 The Knowledge Base

An ontology is a representation of knowledge of a specific domain, done in a machine readable language [2]. The ontology permits to create the domain knowledge base and perform semantic queries on it. The ontology used by the Reachability Matrix Correlator represents connectivity, Network Inventory, Access Control Policy. The ontology is imported from an external file and stored in the Graph Database Sesame [1].

First of all, in order to design the ontology used within the RMC component, we faced a thorough and deep study of IP networks in order to identify all objects that come into play in such a domain, with all their characteristics, the relationships between them, and the role each element plays in successful communications over IP networks. This study has gone in parallel with the analysis of the Data Model that we have been provided with, with special regards for the schemas concerning the Network Inventory and the Deployed Access Policies (which are basically the format of the data incoming into our component as its input), and the schema for the Reachability Matrix which determines the format for the data that exits our component. The resulting knowledge representation, which is an OWL [3] ontology, provides a reconciled vision of these partial Data Models, in such a way that the ontology has "room" enough to receive all data from Network Inventory (and the other input files about routing tables, firewall rules and NAT rules) so as to compose a Knowledge Base (static T-box, plus the A-box reloaded over time), and to re-model the data so as to fit the output data format required by the Reachability Matrix data model.

Of course, input and output formats described in the Data Model do overlap, since are different views of the same matter. More precisely, the output that we produce contains a subset of all the information which is in the input that we receive, but enriched with some "new" information. It is the information made explicit by automatic reasoning, thanks to the "logical embedding" of the additional knowledge about the functioning of IP networks (derived from our initial study) that is recorded in the ontology (in particular in the T-box).

While designing the ontology, the classes described in the Data Model schemas, with all their attributes, needed to be re-modelled to fit the different representation paradigm of ontologies. The most typical cause of intervention is the need to distinguish objects - and their relationships with other objects (possibly with different types of objects) - from simple values that express attributes of the objects (sort of terminal, minimal points of information about which is not possible to say anything else). Briefly, at the present stage of development, the ontology has an expressivity well within OWL-DL [3] expressivity (allowing for good performance of reasoning). It counts with 37 named classes (i.e. concepts in the ontology T-box) that collect the objects accounted for in the Network Inventory and the other input files. There also 37 different relationships (object properties according to the OWL terminology) to represent the possible relationships among the objects of this classes, and other 55 (datatype) properties to account for all other characteristics of the objects.

The most important part for the function of our module (which at present is focused on reachability at the layer 3 of the OSI model) is the part that accounts for: nodes identification, network interfaces, network they belong to, and routing instructions to reach other networks,

i.e. the routes and the complex information to describe them: the source (node and interface), the destination (network), and the gateway to pass through.

Besides the classes that collect the objects of these various types, a set of 12 object properties allow to logically model the reachability between nodes. These (object) properties deal with (a) the network interfaces belonging to some node, (b) the network that each interface is connected to, and (c) the networks that a node belongs to. Furthermore they deal with the other networks that can be reached by passing through one or more gateways, based on routing instructions. Finally, a set of 4 SWRL [4] (Semantic Web Rule Language) rules "force" the reasoner [5] to compute, for every interface of a node, every other node it can reach to (further details on this regard in the next section).

A.3 About the Reasoning

The very first reasoning service used with regard to our ontology is the consistency check of the T-box, sort of a logical validation of the T-box, which is run at the design time of the ontology. Of course, the ontology passed this check. A subsequent check is the validation of the entire knowledge base. Once the A-box is loaded along with the T-box, and the whole KB is loaded into the framework of our component, this second service checks whether the A-box—produced based on the input data (Network Inventory and other files)—is consistent with respect to the T-box. Normally, a fail in this check would highlight an error in the way the input data is translated into the A-box, hence still an error in the ontology design. The most interesting part is the reasoning triggered by the SWRL rules that rely on information stored in the Knowledge Base. These rules are typical logical rules of the form: *If* $condition_1$ *and* $\ldots$ $condition_N$ *THEN consequence*.

The following rules provided along with our ontology describe all possible scenarios to be investigated in order to detect all the nodes that are reachable from any given couple made of a node (with its specific routing instructions) and any of its network interfaces (connected each to one particular network).

Rule 1 covers the case of all reachable nodes within the same domain to which a given node belongs.

Rule 2 covers the case of all reachable nodes within some known networks for which special routing instructions are given.

Rule 3 covers the case of all other networks not known in advance, yet reachable through a series of "hops" to default gateways.

Rule 4 enforces the reasoning in such a way that the transitivity of the relevant relationships (object properties in the ontology describing the functioning of "hopping" through gateways) is properly taken into account by the reasoner [5].

The execution of the reasoning based on all the information within the Knowledge Base and the four SWRL rules, allows to produce the set of all pairs made of a network interface and

the nodes it can reach (discovered by looking at every network interface of a node, its direct connections and the routing instructions given to the node it belongs to). Here we have all information needed to produce the reachability matrix (as it is at present stage, at layer 3 of the OSI model).

Last step to produce our output—the Reachability Matrix—for use on the part of the other components is to explicitly point out, for each network interface of any given node, the set of all and only the other nodes that it can reach. However, this is not properly speaking reasoning, since it is just retrieval of triples (the form in which data are declared in OWL), and it is achieved by firing some SPARQL [6] queries (actually embedded in the APIs [7] of the persistence environment that we adopt). Other similar queries retrieve the rest of information that is available in the Knowledge Base and is expected in the Reachability Matrix according to the output format.

Bibliography

[1] Sesame (http://rdf4j.org/)

[2] Gruber, T. R. (1995). *Toward principles for the design of ontologies used for knowledge sharing*. International journal of human-computer studies, 43(5), 907-928.

[3] McGuinness, D. L., & Van Harmelen, F. (2004). *OWL web ontology language overview*. W3C recommendation, 10(10), 2004.

[4] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). *SWRL: A semantic web rule language combining OWL and RuleML*. W3C Member submission, 21, 79.

[5] Haarslev, V., & Möller, R. (2001). *RACER system description*. In Automated Reasoning (pp. 701-705). Springer Berlin Heidelberg.

[6] Prud'Hommeaux, E., & Seaborne, A. (2008). *SPARQL query language for RDF*. W3C recommendation, 15.

[7] Stellato A. *OWLART API* (http://art.uniroma2.it/owlart/).

B EVENT PRIORITIZATION AND CORRELATION BASED ON PATTERN MINING TECHNIQUES

Notes *This article has been accepted at ICMLA 2015: 14th International Conference on Machine Learning and Applications Workshops, Miami (Florida), December 9-11, 2015.*

Abstract

With the growing deployment of host and network intrusion detection systems in increasingly large and complex communication networks, managing low-level events from these systems becomes critically important. A network has multiple tasks, which consist of multiple network services aiding the execution of a task. An emerging track of security research has focused on event prioritization and correlation to rank the criticality of events and reduce the number of low-level events. To prioritize and correlate events, the ongoing tasks in an enterprise network are identified, as the goal of network operators is to protect ongoing tasks when a security breach occurs. The prioritization of an event depends on the criticality of an ongoing task that is potentially threatened by the event. Additionally, in order to support network operators, we correlate all events that target the same task. A particular task may depend on multiple network services and involve multiple network devices.So, if one network service becomes unavailable, other network services will be affected over time since they are dependent on one another. Unfortunately, dependency details are often not documented and are difficult to discover by relying on human expert knowledge. In order to solve this problem, a network dependency analysis based on network traffic is conducted. We rely on pattern mining techniques to discover tasks in a monitored enterprise network. A formal description of the identified tasks is provided and events are prioritized and correlated based on this model. The pattern mining based network dependency analysis algorithm is evaluated based on a real-world network and three networks that were created with a network simulator.

B.1 Introduction

Malicious actors exploiting cyberspace have been identified by the United States intelligence community as the top national security threat [Cla14]. Similarly, Dell's annual threat report states a 100% increase in SCADA attacks [Del15]. This report is based on an analysis of data gathered by Dell's global response intelligence defense network that consists of millions of security sensors in more than 200 countries. Due to the rising of cybercrime, there has been a great demand for information security systems, e.g., intrusion detection systems.

In the following we understand an Intrusion Detection System (IDS) as a system, which has been developed to monitor an enterprise network for malicious events. An IDS can either be a software or hardware-based system and resulting information is referred to as events or alerts. Generally, these systems are positioned to detect cyber attacks originating from outside a company's own network. IDSs are placed in positions where high volumes of network traffic occur as they can only report on malicious

activity that they are able to observe. To monitor network traffic these systems generally are based on a low-level network traffic model. Thus, low-level events events are created due to the low-level of data that IDSs analyze. Knowledge of both networking technology and infiltration techniques is required to correctly interpret high amounts of high paced low-level events. Furthermore, IBM's 2015 cyber security intelligence index reveals that approximately half of all cyber attacks originate from within a company's own network [Cor15]. Cyber attacks means in effect that at least one network service was exploited.

We depend on network services in many aspects of our daily lives (e.g., email, smart homes, medical services, electricity supply, water supply). Network services operate on distributed sets of clients and servers and rely on supporting network services, such as Kerberos, Domain Name System (DNS), and Active Directory. Hence, network services need to interact with each other in order to function correctly. Since, engineers use the divide-and-conquer approach to implement a new task, they are able to reuse network services and do not need to re-implement complex customized ones. Unfortunately, implementation and dependency details are often not documented, and are difficult to discover by referring to human expert knowledge, for obvious reasons.

Currently, conventional network security approaches focus on perimeter protection instead of identifying the most business critical assets and protect those. Stuxnet [LP13] or Flame [MR12] have taught us that in order to protected critical infrastructures against these advanced persistent threats, perimeter protection simply is not enough. To underpin this statement we refer to the director Sean McGurk of the National Cybersecurity and Communications Integration Center (NCCIC) at the Department of Homeland Security [Sub11]:

"In our experience in conducting hundreds of vulnerability assessments in the private sector, in no case have we ever found the operations network, the SCADA system or energy management system separated from the enterprise network. On average, we see 11 direct connections between those networks. In some extreme cases, we have identified up to 250 connections between the actual producing network and the enterprise network."

IDS events can only be prioritized if the implications of a network service or network device failing are understood. For example an IDS event that is linked to a critical network service needs to be prioritized over an event that is linked to a network service of little importance. To monitor the status of a BP based on all incoming IDS events, all events that target involved network services need to be correlated. In the content of this work we conduct a network dependency analysis based on network traffic to identify activity patterns that we denote as tasks. We formally describe our notion of a task, before prioritizing and correlating events based on task data structures. An evaluation of the identified tasks is based on a real-world network and three networks that were created with the network simulator ns-3 [AMZ12].

B.2 Related Work

Most approaches correlate intrusion events and extract attack scenarios to predict the next attacker action [VS01, FAK15, GGB15, CM02, PZX+08]. However, different intrusion event approaches have different focuses. Several researches have concentrated on how to reduce the amount of events as well as decreasing the false positive rate [MMDD02, Pie04, SJDM08]. Others cluster similar events [CM02, PZX+08] to discover high-level attack scenarios or represent and reason with operator preferences regarding the events they analyze [TBLM11]. Others focus on solving the problem of aggregating events into multi-step attacks as a data mining problem [FAK15, GGB15]. Regardless of their slightly different research focuses, event correlation approaches can be separated into three different

categories: Similarity-based, statistics-based and knowledge-based approaches. Similarity-based event correlation approaches rely on features to compare the similarity of two events or a single event with a cluster of events. Statistics-based security focused approaches [CM02, MMDD02, SJDM08, PZX+08] focus on clustering similar events. Knowledge-based event correlation methods rely on formal knowledge representations to provide predicate-based event correlation. A common drawback of these approaches is that they rely on a lot of human input [CM02, MMDD02, NCR02]. As the complexity of computer networks is growing faster than the ability to understand them, the quality of the human input is questionable. Statistics-based event correlation methods depend on underlying attribute distributions (such as Gaussian distribution) of deviations from what is expected. Valdes et al. [VS01] use statistical methods on multiple event metrics to establish a lower level of correlation before discovering attack step correlations. Farhadi et al. [FAK15] studied how to correlate event patterns using Hidden Markov Models and focused on machine learning based attack plan recognition to correlate and predict events. Unfortunately, most of the presented approaches implicitly rely on a threat agent model [VS01, FAK15, GGB15, CM02, PZX+08]. However, threat agents range from individual hackers, to organized hacker groups, organized crime, industrial espionage, disgruntled employees, terrorist groups to nation state attacks. Due to the lack of information on these different threat agents, predicting their next possible action and validating the proposed adversary model is very difficult. Rather than focusing on how events affect ongoing tasks in the monitored network, the above mentioned approaches focus on aggregating similar events or extracting attack scenarios to predict the next attacker action. Evaluating the quality of a threat model is difficult as little information is available about different threat agents. However, network operators know the purpose of their network and monitored networks provide a multitude of information available for data mining problems. Among the available information is network traffic, routing tables, Network Address Translation (NAT) rules and events. This information can be used to exploit pattern mining approaches to discover tasks in the monitored network. The focus of the approach introduced in the context of this work is to monitor how events affect ongoing tasks in the monitored network. The reasoning behind this approach is that events affecting business-critical tasks have more impact in the eyes of a network operator than others. Also, network operators can judge whether they agree with the identified tasks. Due to little information being available about threat agents, we argue that it is very difficult to conduct a similar evaluation with attacker model based approaches. In contradiction to knowledge-based approaches we focus on minimizing the required human input. The approach introduced in the following is able to (i) automatically discover activities in the network and (ii) correlate incoming events based on targeted tasks.

B.3 Network Model

Modeling an IT network requires a basic understanding of the Open Systems Interconnection (OSI) model. For understanding network connectivity, the following layers of the OSI model are of particular interest: data link layer, network layer, transport layer and application layer. We define a network device as a physical device on the network. The data link layer physically links network devices using MAC addresses to identify devices. However, the data link layer only provides point-to-point connectivity. For enabling network connectivity beyond a point-to-point communication, a network layer protocol such as the Internet Protocol (IP) is required. IP addresses are used to identify source and destination of an end-to-end connection. In other words, MAC addresses allow a point-to-point connection, while IP addresses provide an end-to-end connection. Therefore, switches and routers are used to forward packets, i.e. they act as intermediate hosts.

Definition 1 (Device)**.** *Let MAC and IP be non empty sets of MAC and IP addresses, respectively $(MAC \cap IP = \emptyset)$, then $D \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP)$ is the set of (network) devices.* ▲

Definition 1 allows a device to be assigned multiple MAC addresses and IP addresses. Being able to assign multiple MAC addresses to a network device is needed as routers and switches supply multiple point-to-point endpoints. However, switches do not necessarily need to have IP addresses as they work on the data link layer. From this it follows that they are not visible on the network layer.

Definition 2 (Network)**.** *A (communication) network N is tuple (D, E) consisting of a finite set of devices D and a finite set of edges E, along with $E \subseteq \{\{d_i, d_j\} | d_i, d_j \in D, d_i \neq d_j\}$.* ▲

The end-to-end connection between network devices is the basis of this network model and can automatically be derived based on routing tables or router configuration. Due to the derivation of the network formally introduced in Definition 2, data link layer devices such as switches are opaque. In Example 1 we introduce a running example based on a power grid's communication network and illustrate the corresponding network in Figure B.2.

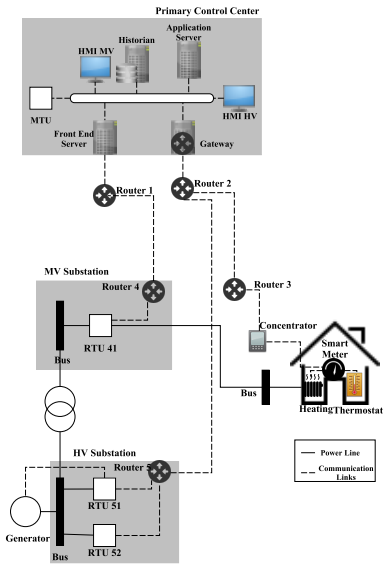


Figure B.1: Running example of a communication network in a power grid.

Example 1. *The schematic diagram in Figure 1 illustrates an excerpt of a power grid network. A primary control center hosts a Human Machine Interface for Medium Voltage Substation (HMI MV) and an HMI for High Voltage Substation (HMI HV). A so called Historian stores historical event and measurement data, and an Application Server analyzes the status of the monitored power grid. Remote Terminal Units (RTUs) collect data from sensors and control actuators situated at remote sites and send data to a Master Terminal Unit (MTU) through the network. The remote sites consist of High Voltage (HV) and Medium Voltage (MV) Substations. Buses are connection points and within the running example they interconnect a generator that is located within the HV Substation via a tranformator and a MV Substation to a "smart home". The "smart home" incorporates a smart meter, intelligent heating and a thermostat. Figure B.2 shows the corresponding network of the schematic diagram introduced in Figure 1. The network is represented as an undirected graph as introduced in Definition 2. Network devices are represented as nodes and communication links correspond to edges.* ♦

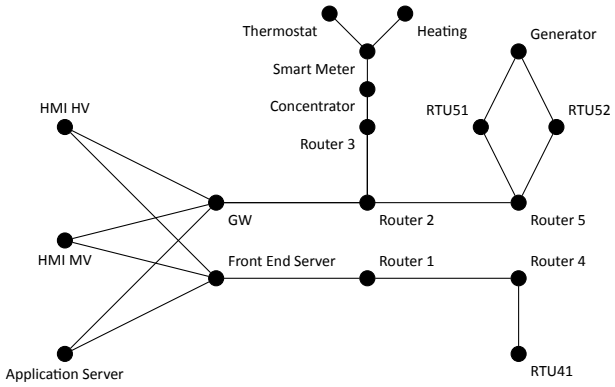


Figure B.2: A small excerpt of the network described in Example 1.

Generally, devices that are end points in your network can host network services. An application can rely on multiple network services to achieve a certain goal. However, a single network service can unambiguously be assigned to an application given that it is not running on a dynamically assigned port. In this paper, we do not make a formal distinction between network services and applications, and we use the term network services. These network services are located on the application layer and are able to communicate end-to-end with each other through a transport layer protocol.

Definition 3 (Network Service)**.** *To derive all network services hosted by a device d_j, we define the relationship $HOSTS(d_j)$, which returns all network services hosted by d_j. In order to derive the device a service s is hosted by, we write $HOSTS^{-1}(s)$, and associate service s_i with device d_j by writing s_i^j. Given a service $s_i^j \in S$, the corresponding device d_j can be derived by $d_j = HOSTS^{-1}(s_i^j)$.* ▲

Below we show how to identify network dependencies based on the presented network model.

B.4 Network Dependency Analysis

Tasks consist of multiple network services that depend on each other to fulfill a common goal. Network services can be directly or indirectly dependent on each other. In order to automatically derive tasks, we need to identify direct and indirect dependencies between network services. Thus, we design a network dependency analysis approach to identify direct and potential indirect dependencies between network services in data networks. The aim of network dependency analysis is to find typical sequences of operation in data networks. A network captures devices that are communication endpoints and additional intermediate devices, over which endpoints communicate. Network devices that are endpoints can host network services. Based on network traffic analysis we will detect network services and determine how they communicate in an end-to-end manner with each other. We conduct a network dependency analysis based on packet headers (e.g. IP, UDP and TCP) and timing data in network traffic.

B.4.1 Direct Dependency

Data is exchanged in the form of packets between the network service s_i^j and s_k^l. We term these end-to-end interactions between network services as direct dependencies. The direct dependency between network services s_i^j and s_k^l is denoted as

$$SDEP = \{(s_i^j, s_k^l) \mid s_i^j \text{ sends a packet to } s_k^l \text{in the period under consideration}\} \quad \text{(B.1)}$$

Based on Definition 3 a direct dependency between network services s_i^j and s_k^l leads to a direct dependency between the respective hosts d_j and d_l. This can be written as

$$DDEP = \{\Big(HOSTS^{-1}(s_i^j), HOSTS^{-1}(s_k^l)\Big) \mid (s_i^j, s_k^l) \in SDEP\} \tag{B.2}$$

Definition 4 (direct dependency)**.** *Let a direct dependency between network services be denoted as SDEP and defined in Equation B.1. We write $\delta(s_i^j, s_k^l)$ to denote $(s_i^j, s_k^l) \in SDEP$.* ▲

Based on Definition 3, a direct dependency between network services results in a direct dependency between devices. We write $\delta(HOSTS^{-1}(s_i^j), HOSTS^{-1}(s_k^l)) = \delta(d_j, d_l)$ to state the derivation of $DDEP$ based on $SDEP$. Over time, network services that are directly dependent exchange packets. A direct dependency $\delta(s_i^j, s_k^l)$ means that a network device d_j hosts a service s_i^j that sends data to another service s_k^l, which is located on network device d_l. This means s_k^l depends on service s_i^j and a delay, disruption, degradation or failure in service s_i^j will lead to delay, disruption, degradation or failure of service s_k^l. Packets include timestamps that denote when the package was exchanged between the two hosts d_j and d_l via the network service s_i^j and s_k^l. A continuous timeline is monitored for finite period of time, which is sliced into k timesteps of a predefined size. Predefined time steps can for example be seconds, milliseconds or nanoseconds. Based on this model we can see when data was exchanged and are able to count the number of packets exchanged within a time step. The packages exchanged between directly dependent network services $\delta(s_i^j, s_k^l)$ over time constitute a vector from $\mathbb{N}^k$. We use $\delta(s_i^j, s_k^l)$ in a predicative way as well as for denoting time series, hence we write $\delta(s_i^j, s_k^l) \in \mathbb{N}^k$.

B.4.2 Indirect Dependency

Beyond direct dependencies there exist more complex dependencies in an IT network. Estimating complex dependencies with $SDEP^+$ would derive complex dependencies based on device connectivity. This would overestimate the number of indirect dependencies and forgo the derivation of a deeper semantic understanding of complex dependencies in a network. To derive a deeper semantic understanding of complex dependencies in a network, we analyze communication patterns and derive indirect dependencies based on "similar" communication patterns. Given a direct dependency $\delta(s_i^j, s_k^l)$, all network services hosted by $HOSTS^{-1}(s_k^l) = d_l$ are candidates for an indirect dependency. An indirect dependency implies that the data received by s_k^l is processed on device d_l. Then, data is sent to another network service s_m^l. Due to the processing of data on device d_l, the request might be sent to s_m^l τ_{delay} time steps later. Hence, the communication patterns of both direct dependencies would be similar, although shifted by τ_{delay} time steps.

Example 2. *Consider the schematic diagram of a network in an electrical power grid presented in Figure B.2 with an operator using HMI MV. The operator intends to send a request to a substation's Remote Terminal Unit (RTU). Hence, an HMI MV sends this request to the Front End Server (FES). The FES transfers the request to a router, which transmits the request to the corresponding router of the substation. The substation's router then sends the request to the RTU. Based on the assumption that communication is based on TCP/UDP, the involved network devices send and receive requests via hosted network services. Problems at any of these involved network services may lead to a failed request for the operator, namely the request not being sent. Operators need to understand the respective indirect dependencies in order to locate the reason why the task "send request to RTU" failed. Also, indirect dependencies help an operator understand the impact of a failed task. If the task "send request to RTU" involves a highly critical device, then a failure of the task could be critical, too.* ♦

In order to understand the general case, we consider two direct dependencies $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$ that are hosted by network devices d_j, d_l and d_o. In some sense, the network services are adjoined.

$$ISDEP = SDEP \bowtie_{HOSTS^{-1}(2)=HOSTS^{-1}(1)} SDEP \tag{B.3}$$

If a network service s_i^j sends a request to s_k^l, which processes the data on device d_l, and an application on d_l sends a request via network service s_m^l to s_n^o, then we refer to the two direct dependencies $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$ as indirect dependency $(s_i^j, s_k^l, s_m^l, s_n^o)$. For brevity, we denote this indirect dependency as $\iota(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))$. Note that network service s_k^l and s_m^l are both hosted by device d_l, i.e. $HOSTS^{-1}(s_k^l) = HOSTS^{-1}(s_m^l)$. Based on Definition 3 an indirect dependency $ISDEP$ between network services leads to an indirect dependency $IDDEP$ between the involved devices. The relation $IDDEP$ is defined as

$$IDDEP = map(\lambda\left((s', s'', s''')\right).\left(HOSTS^{-1}(s'), HOSTS^{-1}(s''), HOSTS^{-1}(s''')\right), \prod_{1,2,4} ISDEP) \tag{B.4}$$

The assumption of indirect dependencies is that data being exchanged within a direct dependency $\delta(s_i^j, s_k^l)$ is also used by a direct dependency $\delta(s_m^l, s_n^o)$. Hence, the two direct dependencies are dependent on each other. Consequently, a failure, degradation or delay of $\delta(s_i^j, s_k^l)$ can lead to failure, disruption or degradation of the other direct dependency $\delta(s_m^l, s_n^o)$ and vice versa. Additionally, all direct dependencies are also described by communication patterns. Example 2 describes a scenario containing such indirect dependencies.

Normalized cross-correlation [BH01] according to Equation B.5 is used to detect how similar both communication patterns $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$, which are both vectors of length k, are. Equation B.6 is used to detect the temporal shift between the $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$. The normalized cross correlation $\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))$ is defined as

$$\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau] = \frac{1}{k}\sum_{n=0}^{k} \frac{\left(\delta(s_i^j, s_k^l)[n] - \overline{\delta(s_i^j, s_k^l)}\right) \cdot \left(\delta(s_m^l, s_n^o)[n+\tau] - \overline{\delta(s_m^l, s_n^o)}\right)}{\sigma_{\delta(s_i^j, s_k^l)} \sigma_{\delta(s_m^l, s_n^o)}}, \tag{B.5}$$

where $\sigma_{\delta(s_i^j, s_k^l)}, \sigma_{\delta(s_m^l, s_n^o)}$ are the standard deviations and $\overline{\delta(s_i^j, s_k^l)}, \overline{\delta(s_m^l, s_n^o)}$ are the means of $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$, respectively.

The point in time τ_{delay}, where both signals are best aligned can be found with

$$\tau_{delay} = \text{argmax}_{\tau \in \{0,\ldots,k\} \subseteq \mathbb{N}} \rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau]. \tag{B.6}$$

If $\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau_{delay}] > \theta$, we consider both communications to be correlated and therefore indirectly dependent and shifted by τ_{delay}. The delay τ_{delay} is added to every indirect dependency in $ISDEP$.

B.4.3 Clustering Dependencies

Aside from intermediate devices (e.g. routers and switches), network devices can be categorized into client and server network devices. In the following we will refer to client network devices as clients and

server network devices as servers. Clients and servers are able to send requests. Servers additionally provide network services that answer these requests. Generally, the number of clients by far surpasses the number of servers. Requests are often sent through a dynamically assigned port. These ports are in the range from $2^{15} + 2^{14}$ to 2^{16} and are available for private, customized or temporary purposes.

Definition 5 (Cluster Network Service)**.** *Let S be a set of services that are hosted by device d_j. All network services communicating through a dynamically assigned port, are grouped by $s^j_* \in S$, which is called cluster network service.* ▲

B.4.4 Tasks

Indirect dependencies are elementary building blocks for deriving tasks in communication networks. We consider indirect dependencies as the smallest possible task, because according to our communication approach a similar communication pattern is detected. Hence, we conclude all network services within the involved direct dependencies are dependent on each other. An indirect dependency $\iota(\delta(s^j_i, s^l_k), \delta(s^l_m, s^o_n))$ contains the notion that a failure or delay of $\delta(s^j_i, s^l_k)$ leads to a failure or delay of $\delta(s^l_m, s^o_n)$. Tasks broaden this notion and cluster all network services that might lead to or be affected by a delay or failure. Hence, the set of tasks TS is defined as

$$TS = SCC\left((S, map(asSet, ISDEP))\right), \tag{B.7}$$

where *SCC* denotes the strongly connected components (SCC) of the hypergraph given as parameter (*asSet* maps a tuple into a set of components). Figure B.3 illustrates a possible set of tasks TS. To find

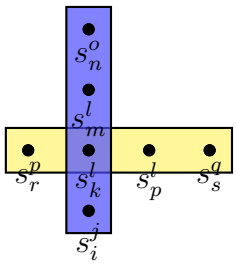


Figure B.3: An example of a set of tasks TS.

the devices TD associated to a task, we use $dev : \mathbb{P}(S) \to \mathbb{P}(D)$ as

$$dev(t) = map\left(HOSTS^{-1}, t\right) \tag{B.8}$$

B.5 Event Processing

After having conducted a network dependency analysis, the focus of this section is to use previously identified network dependencies for event processing. Incoming event streams are heterogeneous data streams. Hence, events have to be normalized before prioritizing and correlating them.

B.5.1 Event Normalization

Based on syslog-ng [Bal15], we integrate events generated by heterogeneous IDS sensors into a standardized data formats called Intrusion Detection Message Exchange Format (IDMEF) [DCF07]. IDMEF was the first attempt to address the problem of formatting and standardizing events [DW01]. The IDMEF data model is a standard representation of events with a predefined set of attributes.

B.5.2 Event Prioritizing and Correlation

Considering a set of events E, every event can be linked to a set of network services according to Equation B.9,

$$ES : E \rightarrow \mathbb{P}(S), \tag{B.9}$$

with a mapping function ES, an event set E and a network service set S. Based on the identified set of network services, a set of involved devices can be derived with the function dev.

Events that affect the same devices are correlated into a single IDMEF event. Whether two events $e_x, e_y \in E$ are correlated is derived by,

$$\begin{aligned} \forall e_x, e_y \in E : ES(e_x) \cap ES(e_y) \neq \emptyset \\ \implies \{e_x, e_y\} \in \textit{IDMEFEV} \end{aligned} \tag{B.10}$$

Correlated events $CORREV$ are defined as the SCCs of the graph (S, *IDMEFEV*). In order to support an operator in understanding how a correlated event affects the monitored network, we map events to a set of devices.

$$\begin{aligned} &\textit{affectedDevices} : E \rightarrow \mathbb{P}(D) \\ &\textit{affectedDevices}(e) = \bigcup_{\substack{t \in TS, \\ \tilde{e}_{xy} \in CORREV}} \{dev(t) \mid ES(\tilde{e}_{xy}) \cap t \neq \emptyset\} \end{aligned} \tag{B.11}$$

An operator is then able to see all events and all devices, which might potentially by affect them. All network devices are assigned criticality values according to the following equation.

$$CRIT : D \rightarrow \{\text{low, medium, high}\} \tag{B.12}$$

Based on the criticality map for devices, we are able to prioritize correlated events by defining the following order relation on *CORREV*:

$$\begin{aligned} \leq = \{(e_x, e_y) \in \textit{CORREV} \times \textit{CORREV} \mid \\ reduce(\text{max}, map(CRIT, \textit{affectedDevices}(e_x)), low) \\ \leq reduce(\text{max}, map(CRIT, \textit{affectedDevices}(e_y)), low)\} \end{aligned} \tag{B.13}$$

A correlated event is ranked as the highest criticality category of all affected network devices. So, a correlated event is assigned a criticality value and it can be ranked into one of the three criticality categories $\{$low, medium, high$\}$. This allows a network operator to prioritize events that might potentially affect more critical network devices.

B.6 Experimental Evaluation

The purpose of the experimental evaluation is to prove the following Hypothesis 1.

Hypothesis 1. *Normalized cross correlation is able to uncover true positive indirect dependencies between network services.*

According to the definitions presents below we implemented an algorithm, which is implemented in C++ and we use OpenCV [Bra00] to apply normalized cross correlation to communication patterns. We

analyze the implemented algorithm based on a data set from a real-world enterprise network. In order to get a ground truth from the identified indirect dependencies are shown to a network operator. The network operator classify the indirect dependencies as true positive or false positive. Given a listing of the identified indirect dependencies, all of them were classified as true positives. However, giving an exhaustive list of all indirect dependencies in the monitored network was outside the field of knowledge of the network operator.

This is why we explored the possibility of creating a communication networks synthetically in order to have a known ground truth. The underlying structure of our synthetically created communication networks is based on three topologies introduced in the "Guide to Industrial Control System Security" from [SFS11], which is modeled with the network simulator Ns-3 [AMZ12] and we add 10% of indirect dependencies randomly. Implementing the test network gives us the ground truth to evaluate the results of the implemented algorithm.

Table B.1: Results of the analyzed data sets.

data set	network services	number of direct dependencies	number of true positive indirect dependencies	number of false positive indirect dependencies
enterprise network	802	829	103	?
ns-3 network 1	48	144	16	1
ns-3 network 2	54	162	19	2
ns-3 network 3	47	141	15	1

The results of the analyzed data sets is shown in Table B.1 and show few false positives.

B.7 Conclusion

In this paper we proposed an event prioritization and correlation approach. In order to correlate all events potentially affecting the same task, we proposed a approach for identifying a network's underlying tasks based on network traffic. Underlying tasks are identified by comparing communication patterns between network service via normalized cross-correlation.

In this work we formally identify tasks as consisting of multiple network services that depend on each other to fulfill a common goal. Future work includes merging indirect dependencies that have multiple common network services into a single dependency. Also, we plan to investigate communicable formats for these low-level tasks.

Bibliography

[AMZ12] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical report, NDN, 2012.

[Bal15] Balabit. Syslog-ng. https://www.balabit.com/network-security/syslog-ng, 2015.

[BH01] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102. International Society for Optics and Photonics, 2001.

[Bra00] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

[Cla14] James R. Clapper. Statement for the record, worldwide threat assessment of the us intelligence community. `https://www.dni.gov/files/documents/Unclassified_2015_ATA_SFR_-_SASC_FINAL.pdf`, 2014.

[CM02] Frédéric Cuppens and Alexandre Miege. Alert correlation in a cooperative intrusion detection framework. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 202–215. IEEE, 2002.

[Cor15] IBM Corporation. 2015 cyber security intelligence index, july 2015.

[DCF07] Hervé Debar, David A Curry, and Benjamin S Feinstein. The intrusion detection message exchange format (IDMEF). In *IETF*, 2007.

[Del15] Dell Inc. Dell security annual threat report. Technical report, 2015.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, pages 85–103. Springer, 2001.

[FAK15] Hamid Farhadi, Maryam AmirHaeri, and Mohammad Khansari. Alert correlation and prediction using data mining and HMM. *The ISC International Journal of Information Security*, 3(2), 2015.

[GGB15] Mohammad GhasemiGol and Abbas Ghaemi-Bafghi. E-correlator: an entropy-based alert correlation system. *Security and Communication Networks*, 8(5):822–836, 2015.

[LP13] Ralph Langner and Perry Pederson. Bound to fail: Why cyber security risk cannot simply be "managed" away. *Cyber Security Series*, 2013.

[MMDD02] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé. M2D2: A formal data model for IDS alert correlation. In *Recent Advances in Intrusion Detection*, pages 115–137. Springer, 2002.

[MR12] Bill Miller and Dale Rowe. A survey scada of and critical infrastructure incidents. In *Proceedings of the 1st Annual conference on Research in information technology*, pages 51–56. ACM, 2012.

[NCR02] Peng Ning, Yun Cui, and Douglas S Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 245–254. ACM, 2002.

[Pie04] Tadeusz Pietraszek. Using adaptive alert classification to reduce false positives in intrusion detection. In *Recent Advances in Intrusion Detection*, pages 102–124. Springer, 2004.

[PZX+08] Xi Peng, Yugang Zhang, Shisong Xiao, Zheng Wu, JianQun Cui, Limiao Chen, and Debao Xiao. An alert correlation method based on improved cluster algorithm. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, volume 1, pages 342–347. IEEE, 2008.

[SFS11] Kevin Stouffer, Kevin Falco, and Karen Scarfone. National institute of standards and technologies - guide to industrial control systems (ics) security. Technical report, National Institute of Standards and Technology, 2011.

[SJDM08] Reuben Smith, Nathalie Japkowicz, Maxwell Dondo, and Peter Mason. Using unsupervised learning for network alert correlation. In *Advances in Artificial Intelligence*, pages 308–319. Springer, 2008.

[Sub11] Subcommittee on National Security, Homeland Defense, and Foreign Operations. Cybersecurity: Assessing the immediate threat to the united states, 2011.

[TBLM11] Karim Tabia, Salem Benferhat, Philippe Leray, and Ludovic Mé. Alert correlation in intrusion detection: Combining AI-based approaches for exploiting security operators' knowledge and preferences. In *Security and Artificial Intelligence (SecArt)*, page NC, 2011.

[VS01] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Recent advances in intrusion detection*, pages 54–68. Springer, 2001.

C TIME SERIES DATA MINING FOR NETWORK SERVICE DEPENDENCY ANALYSIS

Notes *This article has been submitted to CISIS 2016: 9th International Conference on Computational Intelligence in Security for Information Systems, 19th – 21st October, 2016.*

Abstract

In data-communication networks, network reliability is of great concern to both network operators and customers. On the one hand, customers care about receiving reliable services, on the other hand, for network operators it is vital to determine the network parts impacted by e.g. software vulnerabilities being present on a particular network device. To provide network reliability it is fundamentally important to know the ongoing tasks in a network. A particular task may depend on multiple network services, spanning many network devices. Unfortunately, dependency details are often not documented and are difficult to discover by relying on human expert knowledge. In monitored networks huge amounts of data are available and by applying data mining techniques, we are able to extract information of ongoing network activities. From a data mining perspective, we are interested to test the potential data mining techniques have in real-life applications. Hence, we aim to automatically learn network dependencies by analyzing network traffic and derive ongoing tasks in data-communication networks. To automatically learn network dependencies, we propose a methodology based on the normalized form of cross correlation, which is a well-established methodology for detecting similar signals in feature matching applications. We conduct an empirical validation including a comparative cross evaluation, showing the performance and sensitivity of the proposed approach.

C.1 Introduction

For deriving how susceptible a network is to software vulnerabilities, it is essential to understand how ongoing network activities could potentially be affected. A network is built with a higher purpose or mission in mind and this mission leads to interactions of network devices and applications causing network dependencies. A monitored infrastructure's missions can be derived through human labor, however missions are subject to frequent change and often knowledge of how an activity links to network devices and applications is not available. So we are challenged to automatically derive these missions as network activity patterns through network service dependency discovery. Developing a deeper understanding of network activities allows network vulnerability assessment to analyze the confidentiality, integrity and availability impact of software vulnerabilities on a monitored network.

Our contributions. We present a novel framework for Mission Oriented Network Analysis (MONA). The purpose of the proposed approach is assessing the operational impact of software vulnerabilities on the monitored network. In a monitored network large amount of unlabeled data, in form of network traffic, are available for knowledge discovery. We use network traffic to uncover network dependencies

by developing a deeper understanding of network activities. For uncovering network dependencies, we propose a methodology based on normalized cross correlation, which is a well-established methodology for detecting similar signals in feature matching applications. Based on the network simulator ns-3 [AMZ12], we quantitatively evaluate MONA with respect to other network service dependency discovery methodologies.

C.2 Related Work

To motivate our approach, we provide a background to other network dependency assessment methodologies and illustrate their limitations. Network activities in a data-communication network follow from a network having a higher task. Others refer to a network having a higher task as a mission. The concept of missions is sometimes also referred to as mission-centricity in cyber security.

Mission Impact Modeling. Multiple distinct mission-centric approaches to cyber security have been proposed [dBBCY, GDK09, Jak11, KC13, SO08, MTT+11]. An example for a mission-centric approach is the framework for cyber attack modeling and impact assessment [KC13]. They rely on a mission model for generating attack graphs. We understand a mission as network activities with a common purpose, as illustrated in Figure C.1. So our understanding of what constitutes a mission corresponds to Barreto's [dBBCY]. To know how software vulnerabilities affect a mission, we need to understand

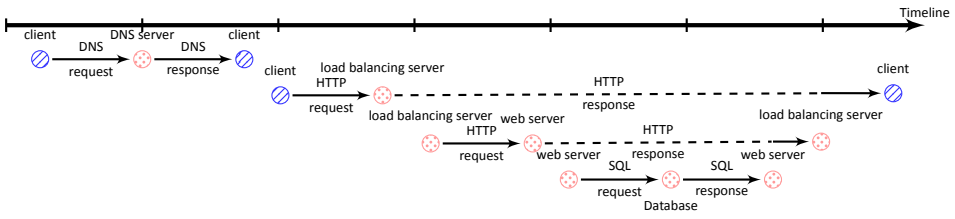


Figure C.1: Example for network activities.

network services dependencies. We say one network service depends on the other if the former requires the latter to operate properly. Automated discovery of network services dependencies from network traffic is discussed in the next paragraph.

Network Service Dependency Discovery. Recent efforts have explored network-based approaches that treat each host as a black box and passively analyze the network traffic between them. For network administrators that are planning to upgrade or reorganize existing applications a dependency discovery approach named Leslie Graph [BBB+06] was designed. The approach aims at identifying complex dependencies between network services and components that may potentially be affected and prevent unexpected consequences. NSDMiner [NNL+12] addresses the same problem of network service dependency for network stability and automatic manageability. Sherlock [BCG+07] is another approach, which learns an inference graph of network service dependency based on co-occurrence within network traffic. A well-known approach is called Orion [CZMB08], which was developed to use spike detection analysis in the delay distribution of flow pairs to infer dependencies. All previously mentioned approaches are based on analyzing network traffic and do not require additional software to be deployed on network devices. Orion, NSDMiner, Sherlock are going to be used later for comparison with MONA.

C.3 IT Network Model

Modeling an IT network requires a basic understanding of the Open Systems Interconnection (OSI) model. For understanding network connectivity, the following layers of the OSI model are of particular interest: data link layer, network layer, transport layer and application layer. We define a network device as a physical device on the network. The data link layer physically links network devices using MAC addresses to identify devices. However, the data link layer only provides point-to-point connectivity. For enabling network connectivity beyond a point-to-point communication, a network layer protocol such as the Internet Protocol (IP) is required. IP addresses are used to identify source and destination of an end-to-end connection. In other words, MAC addresses allow a point-to-point connection, while IP addresses provide an end-to-end connection. Therefore, switches and routers are used to forward packets, i.e. they act as intermediate hosts.

Data-communication networks are built with a common higher purpose. This leads to reoccurring interactions between distinct network devices and services, which we call network activity patterns. An example for such a network activity pattern is given in the following example.

Example for a network activity. An example for a network activity pattern in an IT network is given in Figure C.1. Every node in Figure C.1 represents a network device. Client network devices are represented in blue and server network devices are represented as rose nodes. A majority of communication protocols consist of request and response pairs. Let us assume that a client wants to access a specific web server. Achieving this might require a DNS lookup. Assuming that IP-addresses are returned, a load balancing server receives a HTTP request. The load balancing server passes on the HTTP request a web server. The requested information is not available locally, but it is stored in an external Database. So the web server sends an SQL request to the database. The database sends the information back to the web server. The web server in its turn sends an HTTP response to the load balancing server, which forwards the HTTP response to the client that initiated the sequence of tasks. Per network interface available on a network device, the network device can be connected to another subnetwork. Hence, such a sequence of tasks can require multiple network device and subnetworks to be operational. The purpose of network service dependency discovery is to capture these interactions and how they link to network devices and applications.

Network Device. Let MAC and IP be non empty sets of MAC and IP addresses, respectively $(MAC \cap IP = \emptyset)$, then

$$D^{CY} \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP) \tag{C.1}$$

is the set of network devices.

This allows a device to be assigned multiple MAC addresses and IP addresses. Being able to assign multiple MAC addresses to a network device is needed as routers and switches supply multiple point-to-point endpoints. However, switches do not necessarily need to have IP addresses as they work on the data link layer. From this it follows that they are not visible on the network layer. A network captures devices that are communication endpoints and additional intermediate devices, over which endpoints communicate. Network devices that are endpoints can host network services.

Network service. Let S be a set of network services such that a network service $s_i^j \in S$ is hosted by a network device $d_j \in D^{CY}$. Additionally, the network service is associated by a transport protocol

$\Psi = \{TCP, UDP\}$ and a port. This allows us to define a relation $SERV$, which links a network device, a transport protocol and a port number to a network service by the follow equation

$$SERV : D^{CY} \times \Psi \times \mathbb{N} \rightarrow S. \tag{C.2}$$

To derive all network services hosted by a device d_j, we define the relationship $HOSTS(d_j)$, which returns all network services hosted by d_j. In order to derive the device a service s is hosted by, we write $HOSTS^{-1}(s)$, and associate service s_i with device d_j by writing s_i^j. Given a service $s_i^j \in S$, the corresponding device d_j can be derived by

$$d_j = HOSTS^{-1}(s_i^j). \tag{C.3}$$

Additionally, for a given IP-address and port, we are able to derive the corresponding network device by

$$DEV : \mathcal{P}(IP) \times \mathbb{N}. \tag{C.4}$$

This allows us to derive all involved network services for a given IP-address and port pair by $HOSTS(DEV(sIP, sPort)) \rightarrow \mathcal{P}(S)$. Based on network traffic analysis we will detect network services and determine how they communicate in an end-to-end manner with each other. Aside from intermediate devices (e.g. routers and switches), network devices can be categorized into client and server network devices. In the following we will refer to client network devices as clients and server network devices as servers. Clients and servers are able to send requests. Servers additionally provide network services that answer these requests. Generally, the number of clients by far surpasses the number of servers.

Network Packet. The basic building block of our approach are network packets exchanged between directly dependent network services. A network packet is exchanged by a source and destination IP address srcIP and dstIP via source and destination port srcPort and dstPort. In addition, a network packet relies on a specific transport layer protocol. In the context of this paper we distinguish the transport layer protocols TCP and UDP. We define a network packet as a 6-tuple

$$P = (sIP, sPort, dIP, dPort, \psi, t), \tag{C.5}$$

for source IP addresses sIP, a source ports $sPort$, destination IP addresses dIP, destination ports $dPort$, a transport protocol $\Psi = \{\text{UDP, TCP}\}$ and timestamps t.

Statically and dynamically assigned ports. Requests are often sent through a dynamically assigned port. Dynamically assigned ports are chosen from specifically assigned port ranges [TKL$^+$13]. Ephemeral port ranges are available for private, customized or temporary purposes. Although IANA recommends ephemeral port ranges to range from $2^{15} + 2^{14}$ to 2^{16}, the range is highly dependent on the operation system. Microsoft assigns ephemeral ports starting as low as 1025 for some windows versions and a lot of Linux kernels have the ephemeral port range start at 32768. We follow the IANA recommended ephemeral port range for clustering purposes. Let S be a set of services that are hosted by device d_j. All network services communicating through a dynamically assigned port, are grouped by

$$s_*^j \in S, \tag{C.6}$$

whereas $*$ represents a dynamically assigned port and j represents the device a network service is hosted on. Known network services have to be linked to ports statically, such that other network services can routinely communicate requests with them. It should also be noted that multiple statically assigned ports could be assigned to the same application.

Network Flow. Based on Equation C.5, we conduct a network dependency analysis based on packet headers (e.g. IP, UDP and TCP) and timing data in network traffic. Hence, our approach operates on network flows. To identify network flow boundaries, we look into the definition of TCP and UDP flows. A TCP flow starts with a 3-way handshake (SYN, SYN-ACK, ACK) between a client and a server and terminates with a 4-way handshake (FIN, ACK, FIN, ACK) or RST packet exchange. If network services communicate frequently, they may forgo the cost of repetitive TCP handshakes by using KEEPALIVE messages to maintain a connection in idle periods. In comparison the notion of UDP flows is vague, since UDP is a stateless protocol. This is due to the protocol not having well-defined boundaries for the start and end of a conversation between server and client. In the context of this work, we consider a stream of consecutive UDP packets between server and client as a UDP flow, if the time difference between to consecutive packets is below a predefined threshold. In our analysis we exclude all network packet that are necessary for establishing a communication between server and client. So given that additional data is exchanged between network service s_i^j and s_k^l, we term these end-to-end interactions between network services as direct dependencies. The direct dependency $SDEP$ between network services s_i^j and s_k^l is denoted as

$$SDEP = \{(s_i^j, s_k^l) \mid s_i^j \text{ sends a packet to } s_k^l \text{ in the period under consideration.}\} \tag{C.7}$$

Additionally, we distinguish requests and responses exchanged between network services based on Equation C.6. If a network services uses an ephemeral port to send a network packet to a network service on a static port range, we assume it is a request. Thus, an exchanged request $SDEP^{rq}$ is denoted by

$$SDEP^{rq} = \{(s_*^j, s_k^l) \mid s_*^j \text{ sends a request to } s_k^l \text{ in the period under consideration,}\} \tag{C.8}$$

where k is in the statically assigned port range. Conversely, this means that a network service using its static port range to answer a network service on an ephemeral port is defined as a response. An exchanged response $SDEP^{rsp}$ is written as

$$SDEP^{rsp} = \{(s_k^l, s_*^j) \mid s_k^l, \text{ sends a response to } s_*^j \text{ in the period under consideration.}\} \tag{C.9}$$

C.4 Network Service Dependency Discovery

A data-communication network consists of network devices, which interact due to applications via network services. For example an application "email" uses network services $IMAP$ and $POP3$ to access email messages from a remote network device (i.e. host). From this it follows that an application can rely on multiple network services to fulfill a common goal, which is also referred to as mission. Additionally, we note network activities such as accessing email messages lead to network packets being exchanged by directly dependent network service. The purpose of network service dependency discovery is to abstract network packets in order to detect reoccurring communication pattern. Reoccurring communication patterns indicate that the involved network services are dependent. In order to detect reoccurring communication patterns, we first abstract monitored network traffic into communication histograms.

C.4.1 Communication Histograms

Let us suppose that we are mirroring network traffic from an initial time point t_{min} to a time point t_{max} within an IT network. We are observing network packets $p \in P$, which are defined as a 6-tuple according

to Equation C.5. For communicating network services, we build communication histogram with a bin size Δ_t. In the context of work we set Δ_t to 1 second. The number of histogram bins is given by

$$bins = \lfloor \frac{(t_{max} - t_{min})}{\Delta_t} \rfloor, \tag{C.10}$$

assuming we want to build a communication histogram for network traffic mirrored from time point t_{min} to t_{max} with a bin size Δ_t.

Given that we are monitoring a set of S network services then the data structure for all communication histograms is defined by

$$H : S \times S \times \Psi \rightarrow (\{0, \cdots, bins - 1\} \rightarrow \mathbb{N}_0), \tag{C.11}$$

where the communication histogram bins $\{0, \cdots, bins - 1\}$ are mapped to $\mathbb{N}_0$. Now for every network packet exchanged between directly dependent network services, assuming it was received during the considered time period, the corresponding bin $I(H(s, s', \psi)$ in the communication histogram is incremented. The corresponding bin in the communication histogram is determined by

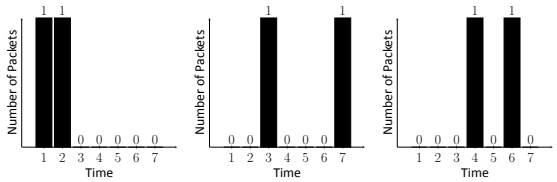
$$(t_{min} - t) \mod bins, \tag{C.12}$$

assuming that the network packet p contains the time stamp t. A very simple algorithm for building communication histograms is described by Algorithm 1.

Algorithm 1 Building communication histograms algorithm

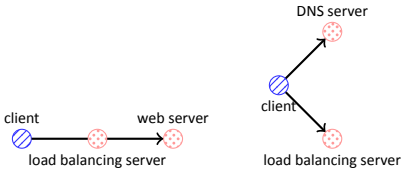
Input:
Network packet **P** $= (sIP, sPort, dIP, dPort, \psi, t)$,
start time $\mathbf{t_{min}}$, stop time $\mathbf{t_{max}}$, time step Δ_t
Output: Communication histograms **H**
▷ compute number of bins for communication histogram **H**
bins = $(t_{max} - t_{min})$ div Δ_t ▷ all histogram vectors are initialized and filled with zeros
▷ fill histogram bins for every network packet in a network flow
for all p = $(sIP, sPort, dIP, dPort, \psi, t) \in$ **P**
and $t \geq t_{min}$ **and** $t \leq t_{max}$ **do**
tb = $(t - t_{min})$ mod bins
H$(SERV(DEV(sIP), sPort),$
$SERV(DEV(dIP), dPort),$
$\psi)[tb]$++
return **H**

Example for communication histograms. To illustrate communication histograms, consider the network activities in Figure C.1. The timeline denotes the chronological sequence of exchanged network packets with the corresponding communication histogram bins. The communication histograms are illustrated in an exemplary manner in Figure C.2. Given that we have now abstracted network traffic into signals, we are now able to look into network service dependency discovery.



(a) Client →DNS Server **(b) Client →Load ancer** **(c) Load bal-bal-ancer → web server**

Figure C.2: Example for communication histograms.



(a) Remote-remote indirect dependency. **(b) Local-remote indirect dependency.**

Figure C.3: Example for indirect dependencies.

C.4.2 Indirect Dependencies

Network services operate on distributed sets of clients and servers and rely on supporting network services, such as Kerberos, Domain Name System (DNS), and Active Directory. To fulfill a network's mission, network services need to interact. Since, engineers use the divide-and-conquer approach to implement a new task, they are able to reuse network services and do not need to re-implement complex customized ones. This leads to multiple network services interacting for a common high-level task. For the purpose of detecting indirect dependencies, we analyze the communication histograms of directly dependent network services in order to derive re-occurring communication patterns. Detecting re-occurring communication patterns requires clustering direct dependencies. Similarly to previous work, we distinguish two different types of remote-remote dependencies and local-remote dependencies [CZMB08]. A local-remote (LR) dependency is an indirect dependency, where a system must issue a request to a remote system in order to complete an outstanding request issued to a local service. A remote-remote (RR) dependency is one in which a system must first contact one host before issuing a request to the desired host.

Example for indirect dependencies. Based on the example of network activities given in Figure C.1, an example for a RR dependency could be: The client is communicating with the load balancing server, who then communicates with the web server. A LR dependency could be: The client resolves a DNS name and then uses the IP-address to contact a load balancing server. Figure C.3 illustrates an example for RR and LR dependencies.

Candidates for Indirect Dependency. Given a direct dependency $\delta(s_i^j, s_k^l)$, all network services hosted by

$$HOSTS^{-1}(s_k^l) = d_l \text{ or } HOSTS^{-1}(s_i^j) = d_j \tag{C.13}$$

are candidates for an indirect dependency. The communication histograms contain the communication pattern of all involved directly dependent network services.

C.4.3 Measuring Communication Histogram Similarity

Suppose we have two communication histograms r and $s \in H$, which are candidates for being indirectly dependent. The two communication histograms are time series $r = (r_1, r_2 \cdots, r_{bin})$, consist of bin samples.

Pearson Correlation coefficient. In statistics, the Pearson product-moment correlation coefficient the linear correlation σ between two variables X and Y by

$$\sigma = \frac{[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \tag{C.14}$$

where μ_Y and σ_Y denotes the mean and standard deviation of Y. The Pearson product-moment correlation coefficient is a measure of the linear correlation between two variables X and Y, giving a value between +1 and -1 inclusive, where

1 is total correlation,

0 is no correlation, and

-1 is total anti-correlation.

Pearson Distance. To apply the Pearson correlation coefficient as a distance measure on a time series [HK09], we define the Pearson Distance as

$$d_\varrho(r, s) = \frac{\frac{1}{bins} \sum_{t=0}^{bins} (r_t - \mu_r)(s_t - \mu_s)}{\sigma_r \sigma_s}, \tag{C.15}$$

where μ_r and σ_r are mean and standard deviation of r, such that $-1 \leq d_\varrho \geq 1$.

Network Latency. An indirect dependency implies that the data received by s_k^l is processed on device d_l. Then, data is sent to another network service s_m^l. Due to the processing of data on device d_l, the request might be sent to s_m^l τ_{delay} time steps later. Network latency can lead to communication patterns being shifted due to the time it takes for a network packet to be transferred. Communication patterns are stored in communication histograms and they contain the same pattern, which thus are also shifted due to network latency. Hence, the communication patterns of both direct dependencies would be similar, although shifted by τ_{delay} time steps. In pattern recognition, normalized cross correlation has been proposed to take a shift, such as τ_{delay}, into account.

Normalized cross-correlation. To overcome the lack of a perfect alignment between two communication networks, we extend the Pearson distance, introduced in Equation C.15, to normalized cross-correlation [BH01].

$$\varrho_{r,s}(\tau) = \frac{\frac{1}{bins}\sum_{t=0}^{bins}(r_t - \mu_r)(s_{t+\tau} - \mu_s)}{\sigma_r \sigma_s}, \tag{C.16}$$

The point in time τ_{delay}, where both signals are best aligned can be found by computing

$$\tau_{delay} = \text{argmax}_{\tau \in \{0,\ldots,bins-1\}} \varrho_{r,s}(\tau). \tag{C.17}$$

If $\varrho_{r,s}(\tau) \geq \theta$, we consider both communication histograms r and s to be correlated and therefore indirectly dependent and shifted by τ_{delay}. Normalized cross-correlation is applied to all indirect dependency candidates and returns a set $ISDEP$, which is derived by applying

$$ISDEP = SDEP \bowtie SDEP \tag{C.18}$$

on all LR (local-remote) and RR (remote-remote) dependencies. The set of indirect dependencies describes re-occurring network activities, so now we need to inquire how software vulnerabilities affect these re-occurring network activities.

C.5 Experimental Evaluation

The disaster recovery site of an energy distribution network, provided an Italian water and energy distribution company, was available for network traffic analysis. Based on this network, we are able to collect and analyze real-life network traffic. Real-life network traffic consists of network services frequently to rarely interacting and we are able to determine typical response times. Some network services have a common purpose and show similar communication patterns. As this network is a real-life network, absolute knowledge of all network dependencies is not available. During first experiments on data sets from the disaster recovery site, we often found new network dependencies that had been previously forgotten by the network operators. As every network relies on third party software, which might have their own network dependencies unknown to network operators, collecting a known ground truth on this network is not feasible. Hence, we extracted the communication patterns of known network dependencies and used this information to develop a random network generator in ns-3 [AMZ12]. This random network generator can create synthetic data sets with a known ground truth for testing our network dependency analysis.

Focus of the evaluation is performance and sensitivity network service dependency discovery. Indirect dependencies are the fundamental building block for the proposed network vulnerability assessment. Given that indirect dependencies are identified correctly, software vulnerabilities are linked to network services and their operational impact is assessed. Hence, the significance of our proposed network vulnerability framework is dependent on precision and recall of network dependency discovery. To compare the precision of the proposed network dependency analysis, we compare our result with respect to other network service dependency discovery methodologies.

Network simulation is widely used to design and evaluate new protocols and applications. The chosen network simulator in the context of this work is ns-3 [AMZ12], and it has been used in numerous domains from simulating peer-to-peer, wireless and ad-hoc networks, business processes or peer-to-peer network.

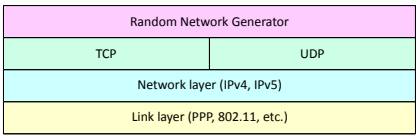


Figure C.4: Network layer model of the ns-3 based random network generator.

Figure C.4 illustrates the network simulation as essentially being a new ns-3 module. The ns-3 module is comprised of a random network generator, which is built to run on top of any available link-layer protocol model/on top of network-layer/transport-layer protocols. For our experimental evaluation, link, network and transport layers are provided by ns-3. We collect typical communication patterns in the PANOPTESEC [Con16] testbed and abstracted the communication delay distributions in order to build a network generator. We added some random variations to the communication delays to mimic noise, which is always present in real-world applications. Based on this information, we randomly generate networks with varying communication patterns and a predefined number of network nodes and indirect dependencies.

C.5.1 Comparable Network Dependency Analyzers

Orion. Orion [CZMB08] infers network dependencies between two network services, if the delay between consecutive access follow a constant pattern. As an example for such constant pattern, consider an application, which needs consecutive access to two distinct network services. Then, the delay between to dependent network services will follow a non-random distribution. It has already been pointed out previously [Mar13] that it contains several fixed constants that are selected without prior data calibration, which might lead to shortcomings in the robustness of the proposed solution. Also, Orion requires a minimum flow count to analyze the delay distribution of network services. Given that enough network flows are provided, Orion calculates a threshold

$$mean + x \cdot stdev \tag{C.19}$$

over the delay distribution. The multiplication factor x is a fixed constant and given that the threshold is surpassed, the corresponding network services are classified as indirectly dependent. Additionally, Orion introduces the terminology of local-remote and remote-remote dependencies and leverages the delay distribution between network services to infer network dependencies.

NSDMiner. NSDMiner [NNL+12] is another methodology for network service dependency discovery. NSDMiner locates nested connection, wherein one complete request-response pair starts and completes between the request and response of another connection. So the located nested connection has to match an expected recursive connection pattern. Network flows are monitored for their chronological order, and NSDMiner detects a network service dependency, if the probability that the life span of connections to one network service is included in the life span of a connection to another network service is higher than a predefined threshold. NSDMiner focuses on detecting local-remote dependencies.

Sherlock. Sherlock [BCG+07] infers network service dependency based on co-occurence within network traffic. Similar to NSDMiner, Sherlock builds on the notion of nested connections. As a result, the strength of a network service dependency is computed as the probability of a network service being

accessed within a time interval in which another network service is accessed. It focuses on detecting remote-remote dependencies and leverages packet capture running at each end-host to infer dependencies. After inferring network service dependencies, a directed dependency graph is built, modeling network device states as up, troubled or down.

C.5.2 Analysis Methods

To allow a comparative evaluation of all network service dependency methods, we randomly generate network traffic with the previously described ns-3 based module. For an evaluation, it is important to have a known ground truth. Ground truth means that each data-communication network contains network traffic between directly and indirectly dependent network services and we have knowledge of all existing network service dependencies. We focused the experimental analysis on determining how correct the learned network service dependency model is in comparison to other network dependency analyzers. As we are able to control the direct and indirect dependencies in the network, we are able to analyze precision and recall of the learned model.

Analysis Method. True Positive (TP) refers to a correctly learned indirect dependency, False Positive (FP) refers to a learned indirect dependency that is false and False Negative (FN) is an existing indirect dependency that was not found. Based on these values we can compute Precision and Recall as normal

$$Precision = TP/(TP + FP), \tag{C.20}$$

$$Recall = TP/(TP + FN).$$

Test environment. The environment used for the evaluation is the following:

- Virtual machine on Macbook Pro, 2,8 GHz Intel Core i7, OSX 10.10.4 (14E46)
- OS:Ubuntu Release 12.04 (precise) 64-bit, Kernel Linux3.13.0-32-generic
- CPU: Intel Core i7-4558U @ 2.80GHz (1 processor)
- RAM: 8 GB, 1600 MHz DDR3

C.5.3 Performance and Sensitivity Evaluation

Based on the previously described random network generator, we are able to conduct experiments to test performance and sensitivity of the network service dependency discovery methodologies. To provide proof that our proposed methodology is sound, we first evaluate the performance with respect to all approaches. Afterwards, a sensitivity analysis will address the limiting factors of our approach.

In order to analyze the quality of all network service dependency discovery methods, we repeatedly generate data-communication networks containing 100 network devices with 30 indirect dependencies. We adjusted the threshold for all methodologies to reflect the best possible result. The results of this evaluation are shown in Figure C.4. Given all indirect dependencies uncovered by MONA, we analyze how many where also detected by Orion, NSDMiner and Sherlock. NSDMiner's performance is partially due to the methodology only detecting local-remote dependencies. Therefore, it misses remote-remote

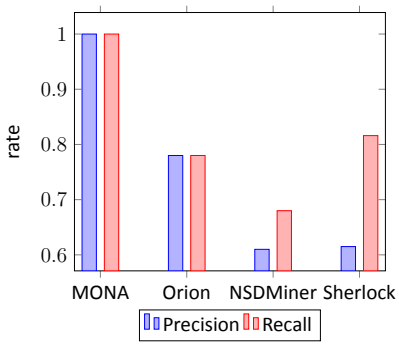


Figure C.5: Comparison between MONA, Orion, NSDMiner and Sherlock.

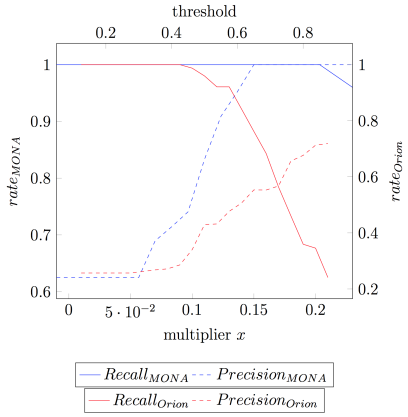


Figure C.6: Comparison between MONA and Orion.

dependencies. Sherlock, among other problems, detects every pair of frequently occurring network services as depending on each other. Thus, it creates a large number of false positives.Similar to Rippler's experiment [ZVKK14], the results of our experiments verified this property.

Sensitivity Analysis. Orion and MONA both rely on a threshold in order to quantify indirectly dependent network services. Orion's threshold is given in Equation C.19, while MONA's threshold is based on Equation C.16. In order to be able to compare both thresholds, we used our random network generator to create data-communication networks containing 100 network devices, 60 directly communicating network services and 20 indirect dependencies. To overcome a possible bias of the results due to the added random variations, we generated every network multiple times and averaged the results. Figure C.6 show the results of this analysis.

Orion. In their own experimental evaluation, Orion has a high rate of TP and a low rate of FN indirect dependencies in data-communication networks with small delays or huge amount of network communication between indirect dependent network services. However, not all data-communication networks fulfill such criteria. To further prove our point additionally created a data-communication networks containing 450 network devices with communication, 100 communicating network services and between 120 and 140 indirect dependencies. The results for this analysis are shown in Figure C.7 and reflect our previous assessment. In data-communication networks of power grids a lot of network devices are geographically distributed. Therefore, network communication delays are more spread out then

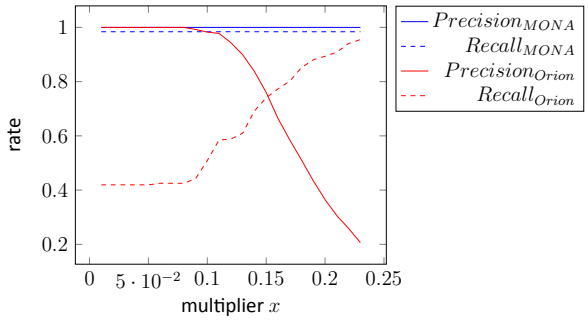


Figure C.7: Results for Orion in a larger data-communication networks.

in data-communication networks without geographically distributed areas. Our observation regarding Orion failing to detect high-confidence dependencies is mirrored by the experimental evaluation of the active network dependency analyzer Rippler [ZVKK14].

MONA. To assess the sensitivity our introduced approach, we generated a network containing 500 network devices and 500 direct dependencies. In other words, this network does not contain any indirect dependencies that could be identified. The results of this analysis are shown in Figure C.8.

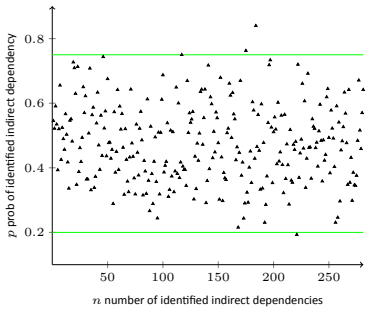


Figure C.8: Results for MONA in a data-communication network containing no indirect dependencies.

Performance Evaluation. Computing network service dependencies and linking these to detected software vulnerabilities is conducted offline. Nevertheless, we need to make sure that network dependency information can be gathered based on practically relevant data for network traffic in realistic scenarios in sensible times. For this purpose, we use synthetic network traffic data obtained in a scalable network (100 devices). Figure C.9 evaluates the performance of network dependency in a network containing 100 devices, 50 communicating network services and a variable number of indirect dependencies.

C.6 Conclusion

We have demonstrated a novel network vulnerability analysis approach. Subsequence mining enables deriving a deeper understanding of network activities and leverages well data-communication networks with different numbers of network devices and indirect dependencies. A systematic evaluation based on the network simulator ns-3, compared the performance and sensitivity of our proposed approach to

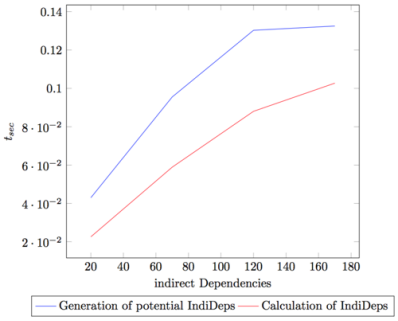


Figure C.9: 100 devices, 50 direct dependencies, variable indirect dependencies

three other network dependency discover methodologies. We extended network dependencies in order to derive the confidentiality, integrity and availability impact of software vulnerabilities.

Bibliography

[AMZ12] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical report, NDN, 2012.

[BBB+06] Paramvir Bahl, Paul Barham, Richard Black, Ranveer Chandra, Moises Goldszmidt, Rebecca Isaacs, Srikanth Kandula, Lun Li, John MacCormick, David A Maltz, et al. Discovering dependencies for network management. In *ACM SIGCOMM 5th Workshop on Hot Topics in Networks (Hotnets-V)*, pages 97–102, 2006.

[BCG+07] Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 13–24. ACM, 2007.

[BH01] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102. International Society for Optics and Photonics, 2001.

[Con16] PANOPTESEC Consortium. Panoptesec. http://www.panoptesec.eu, 2016.

[CZMB08] Xu Chen, Ming Zhang, Zhuoqing Morley Mao, and Paramvir Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *OSDI*, volume 8, pages 117–130, 2008.

[dBBCY] Alexandre de Barros Barreto, Paulo Cesar G Costa, and Edgar T Yano. A semantic approach to evaluate the impact of cyber actions on the physical domain. *STIDS 2012 Committees*.

[GDK09] John R Goodall, Anita D'Amico, and Jason K Kopylec. Camus: automatically mapping cyber assets to missions and users. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE, 2009.

[HK09] Frank Höppner and Frank Klawonn. Compensation of translational displacement in time series clustering using cross correlation. In *Advances in Intelligent Data Analysis VIII*, pages 71–82. Springer, 2009.

[Jak11] Gabriel Jakobson. Mission cyber security situation assessment using impact dependency graphs. In *FUSION*, pages 1–8, 2011.

[KC13] I. Kotenko and A. Chechulin. A cyber attack modeling and impact assessment framework. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–24, June 2013.

[Mar13] Scott Marshall. Candid: Classifying assets in networks by determining importance and dependencies. Master's thesis, University of California at Berkeley, 2013.

[MTT+11] S. Musman, M. Tanner, A. Temin, E. Elsaesser, and L. Loren. Computing the impact of cyber attacks on complex missions. In *Systems Conference (SysCon), 2011 IEEE International*, pages 46–51, April 2011.

[NNL+12] Arun Natarajan, Peng Ning, Yao Liu, Sushil Jajodia, and Steve E Hutchinson. *NSDMiner: Automated discovery of network service dependencies*. IEEE, 2012.

[SO08] Reginald E. Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *Proceedings of the 13th European Symposium on Research in Computer Security*, pages 18–34, 2008.

[TKL+13] Joe Touch, M Kojo, E Lear, A Mankin, K Ono, M Stiemerling, and L Eggert. Service name and transport protocol port number registry. *The Internet Assigned Numbers Authority (IANA)*, 2013.

[ZVKK14] Ali Zand, Giovanni Vigna, Richard Kemmerer, and Christopher Kruegel. Rippler: Delay injection for service dependency detection. In *INFOCOM, 2014 Proceedings IEEE*, pages 2157–2165. IEEE, 2014.

D USING A DEEP UNDERSTANDING OF NETWORK ACTIVITIES FOR NETWORK DEPENDENCY ANALYSIS

Notes *This article is accepted and will be published at KI2016: 39th edition of the German Conference on Artificial Intelligence, Klagenfurt (Austria), September 26-30, 2016.*

Abstract

Workflow mining is the task of automatically detecting workflows from a set of event logs. We argue that network traffic can serve as a set of event logs and, thereby, as input for workflow mining. Networks produce large amounts of network traffic and we are able to extract sequences of workflow events by applying data mining techniques. We come to this conclusion due to the following observation: Network traffic consists of network packets, which are exchanged between network devices in order to share information to fulfill a common task. This common task corresponds to a workflow event and, when observed over time, we are able to record sequences of workflow events. Sequences of workflow events are caused by network dependencies, which force distributed network devices to interact. To automatically learn network dependencies, we propose a methodology based on the normalized form of cross correlation, which is a well-established methodology for detecting similar signals in feature matching applications. Furthermore, by observing network dependencies over time, we are able to automatically detect workflows.

D.1 Introduction

Workflow and business process models have recently gained a lot of traction in the cyber security community. This is due to the fact that they can be used as a foundation for operational impact assessment within a security information and event management system, or as a foundation for process-aware information systems. In the following, we will only use the term workflow, however it should be noted that earlier publications use the terms workflow and business process models interchangeably [Her00]. Companies, organizations and enterprises have a workflow, which translates into network activities within their data communication network. An example for network activities caused by a workflow is given in

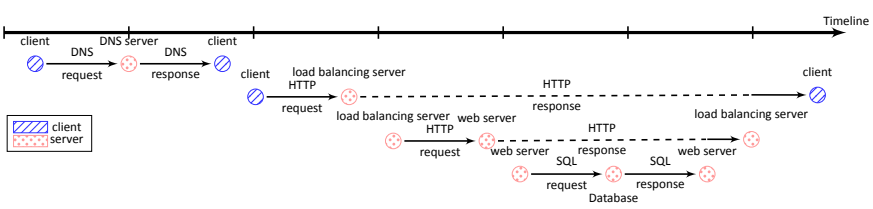


Figure D.1: Example for network activities.

Figure D.1, which shows a customer visiting a web shop and the web shop fetches product information from a database. In Figure D.1 every node represents a network device with clients being represented by blue nodes and servers by rose nodes. Workflows are often not documented and there have been

reoccurring issues [VDAvDH+03] with manually designed workflows. It is a very time consuming process to design hand-made workflow models and, thereby, expensive. Hand-made workflows are idealized descriptions of the process at hand and often describe more what should be done, than the actual process. Additionally, with a hand-made workflow it difficult to detect when concept drifts have occurred and the workflow model needs to be updated. Hence, the data mining community has paid a lot of attention to the automated acquisition of workflow models [Her00, VDAWM04, VDA04, VDVDA04, VDAAdM+11]. Based on event logs, workflow mining methods automatically deduce sequences of workflow events.

In the context of this work we present a novel approach to workflow mining called Mission Oriented Network Analysis (MONA), which is able to deduce a workflow model based on network traffic. The remaining paper is organized as follows: Section D.2 proved a background on related work. Section D.3 introduces network service dependency discovery for detecting workflow events based on network traffic. Section D.4 explains how to derive a HMM based workflow model based on the previously detected workflow events. Then, Section D.5 a systematic evaluation against other network service dependency discover methods is presented.

D.2 Related Work

In the context of this work, we use network traffic to first derive network service dependencies and, subsequently, workflows. Thus, we first provide a background to workflow discovery techniques and then list network service dependency discovery methods. Workflow mining is dependent on event logs and research in this domain can be divided into three topics: discovery, conformance and enhancement of workflows [VDA11]. As we focus on workflow discovery in the context of this work, we list workflow discovery techniques in the following. A lot of workflow mining methods [Her00, VDAWM04, VDA04, VDVDA04, VDAAdM+11] rely on Petri nets, due to their similarities to workflow models established in business science. Other workflow mining methods [SZS05, BPFN08] rely on Hidden Markov Models (HMMs), which are statistical models. Unlike Petri nets, HMMs are able to model properties such as the average duration of a workflow event and the transition probability between workflow events. However, Petri nets can be efficiently mapped to HMMs [RVvdA08, PM16]. HMMs are used for a wide range of applications and efficient techniques exist for modeling a HMM and estimating the likelihood that observed data was produced by this HMM. The latter enables evaluating, when relearning the HMM is necessary and, thereby, counter a possible concept drift. Sequence clustering [FZMF07] deduces a sequence of tasks and groups similar sequences into a common cluster. These sequences of tasks are modeled as a first-order Markov chain and, due to their probabilistic nature, this makes the technique more robust to noise. Trace clustering is another technique, which extracts features from traces and clusters them based on these extracted features [SGVDA08]. A very popular workflow mining technique is the α algorithm [VDA04, VDAWM04], which searches for ordering relations in event logs and models workflows based on Petri nets. For the α algorithm it is proven that for certain subclasses it is possible to find the right workflow model. Not all workflow mining techniques rely on Petri nets. For example, inductive workflow acquisition [HK98] derives a HMM by analyzing event logs and instance graphs [VDVDA04] by aggregating multiple graphs to model a workflow. Event logs are the basis for all previously listed techniques and are supposed to contain workflow data. Obtaining workflow information is not as easy and often experiments are conducted based on synthetic data sets [SZS05]. A common limitation that all previously listed techniques have is that they rely on event logs containing workflow data, a data source that is hard to come by in every day enterprise networks.

However, a data source that is very easy to obtain from a data communication network is network traffic.

Hence, we propose a workflow mining technique, which is able to derive workflows based on network traffic. To achieve this goal, we rely on network service dependency discovery to identify workflow events. Thus, automated discovery of network service dependencies from network traffic is discussed in the next paragraph.

Recent efforts have explored network-based approaches that treat each host as a black box and passively analyze the network traffic between them. For network administrators who are planning to upgrade or reorganize existing applications a dependency discovery approach named Leslie Graph [BBB+06] was designed. The approach aims at identifying complex dependencies between network services and components that may potentially be affected and prevent unexpected consequences. NSDMiner [NNL+12] addresses the same problem of network service dependency for network stability and automatic manageability. Sherlock [BCG+07] is another approach, which learns an inference graph of network service dependency based on co-occurrences within network traffic. A well-known approach is called Orion [CZMB08], which was developed to use spike detection analysis in the delay distribution of flow pairs to infer dependencies. All previously mentioned approaches are based on analyzing network traffic and the presented algorithms do not require additional software to be deployed on network devices. To the best of our knowledge, MONA is the first approach that expands network service dependency discovery to workflow discovery. Orion and Sherlock are going to be used later for a comparative evaluation of MONA.

D.3 Network Service Dependency Discovery

As shown in Figure D.1, workflows often cause reoccuring network activity patterns. We understand these network activity patterns as workflows and network service dependency analysis has the purpose of detecting workflow events.

D.3.1 Network Model

A network device is defined as a non empty set of MAC and IP addresses MAC and IP, respectively $(MAC \cap IP = \emptyset)$, where

$$D \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP) \tag{D.1}$$

is the set of network devices. As previously mentioned, network devices can host network services. Let network services be defined as a finite non-empty set of S, such that a network service $s_i^j \in S$ is associated to a port $p \in PORT$, which is defined as $PORT \subseteq \mathbb{N}$. A network service $s_i^j \in S$ is hosted by a network device $d_j \in D$ and this is formalized as

$$SERV : D \times PORT \rightarrow S. \tag{D.2}$$

The basic building blocks of network traffic are network packets. Let P be the set of all network packets, which are defined as

$$P : SERV \times SERV \times \Psi \times T, \tag{D.3}$$

where a source and destination network service $SERV$ exchange data based on a specific transport layer protocol $\Psi = \{\text{UDP, TCP}\}$ at a certain time point in T. In other words, T is a set of time stamps that is isomorphic to natural numbers $\mathbb{N}$. The set of all network packets P. In the following, we conduct a network dependency analysis and additionally exclude all network packet that are necessary for establishing a communication between server and client.

Additionally, for a given IP address and port, we are able to derive the corresponding network device by

$$DEV : IP \rightarrow D. \tag{D.4}$$

To derive all network services hosted by a device d_j, we define the relationship *HOSTS*(d_j), which returns all network services hosted by d_j. Given a service $s_i^j \in S$, the corresponding device d_j is derived by

$$d_j = \textit{HOSTS}^{-1}(s_i^j). \tag{D.5}$$

Clients and servers are able to send requests, which are sent through dynamically assigned ports, chosen from specifically assigned port ranges [TKL$^+$13]. These dynamically assigned ports often change, so these network services are grouped by $s_*^j \in S$, where $*$ represents a dynamically assigned port. Network services that answer these requests have to be linked to ports statically, such that other network services can easily localize where to send their requests to. These network services are not grouped as these ports are statically assigned and are not subjected to frequent changes. Based on this network model, focus of the next section is to characterize the interaction between network services.

D.3.2 Direct Dependencies

So given that data is exchanged between network service s_i^j and $s_k^l \in S$, we term these interactions between network services as direct dependencies. A direct dependency $SDEP$ between network services s_i^j and s_k^l, is denoted as

$$SDEP = \{(s_i^j, s_k^l) \mid s_i^j \text{ sends a packet } p \in P \text{ to } s_k^l \text{ in the period under consideration.}\} \tag{D.6}$$

Let us suppose that we are analyzing network packets $P_i \subseteq P$, as introduced in Equation D.3 exchanged within a time window $[t_{min}, \ldots, t_{max}]$, with start and end time point $t_{min}, t_{max} \in T$ within an IT network. Network services exchanging these network packets P_i form a set $S_i \subseteq S$. To build a communication histogram H, we set a bin size Δ_t and count the numbers of network packets exchanged between two network services. The number of bins in a communication histogram H is given by $bins = \lfloor \frac{(t_{max} - t_{min})}{\Delta_t} \rfloor$. All communication histograms are defined by

$$H : S \times S \times \Psi \rightarrow (\{1, \cdots, bins\} \rightarrow \mathbb{N}_0). \tag{D.7}$$

In the prior equation, the communication histogram bins $\{1, \cdots, bins\}$ are mapped to $\mathbb{N}_0$. A communication histogram H provides an array, containing the numbers of network packets exchanged in every time stamp Δ_t. For every exchanged network packet, assuming it was received during the considered time period $[t_{min}, \ldots, t_{max}]$, the corresponding bin tb in the communication histogram $H(s, s', \psi)$ is incremented $H(s, s', \psi)[tb] + +$. This is described in Algorithm 2. The corresponding bin tb in the communication histogram is determined by $tb = \frac{(t_{min} - t)}{bins}$, assuming a timestamp $t \in T$.

D.3.3 Indirect Dependencies

In the context of this work, we introduce network dependency analysis as a basis for workflow mining. For this purpose, indirect dependencies correspond to workflow events. Similarly to previous work, we distinguish remote-remote dependencies and local-remote dependencies [CZMB08]. Examples for both dependency types are shown in Figure D.2 and the example is based on network activities described in Figure D.1. For an remote-remote (RR) dependency, first one remote host must be contacted before

Algorithm 2 Building communication histograms algorithm

Input: Network packets $P_i \subseteq P$,
start time $\mathbf{t_{min}}$, stop time $\mathbf{t_{max}}$, time step Δ_t
Output: Communication histograms **H**
bins = $(t_{max} - t_{min})$ **div** Δ_t ▷ all histogram vectors are initialized and filled with zeros
for all $p \in P_i$ **do**
    $tb = (t - t_{min})$ **mod** $bins$
    H$(SERV(DEV(sIP), sPort),$
      $SERV(DEV(dIP), dPort), \psi)[tb]$++
return **H**

issuing a request to another remote host. Figure D.2a shows an RR dependency $ISDEP_{RR}$, which is defined with a theta join

$$ISDEP_{RR} = SDEP \bowtie_{\mathit{HOSTS}^{-1}(2)=HOSTS^{-1}(1)} SDEP, \tag{D.8}$$

where $\mathit{HOSTS}^{-1}(1)$ takes the first network service of a direct dependency $SDEP$ and returns the network device it is hosted on. A local-remote (LR) dependency consists of a system issuing a request to a remote system in order to complete an outstanding request issued to a local service. A LR dependency $ISDEP_{LR}$ is shown in Figure D.2b and is defined as

$$ISDEP_{LR} = SDEP \bowtie_{HOSTS^{-1}(1)=HOSTS^{-1}(1)} SDEP. \tag{D.9}$$

The set of all RR and LR dependencies is defined as $\mathit{ISDEP} = \{\mathit{ISDEP}_{LR}, \mathit{ISDEP}_{RR}\}$. Indirect dependences

(a) Remote-remote indirect dependency. **(b) Local-remote indirect dependency.**

Figure D.2: Example for indirect dependencies.

are derived based on direct dependencies and given $(s_i^j, s_k^l) \in$SDEP, all network services hosted by

$$HOSTS^{-1}(s_k^l) = d_l \text{ or } HOSTS^{-1}(s_i^j) = d_j \tag{D.10}$$

are candidates for a LR dependency $((s_i^j, s_k^l), (s_m^j, s_o^n)) \in$ISDEP. Both (s_i^j, s_k^l) and (s_m^j, s_o^n) are described by communication histograms as introduced in Equation D.7.

Normalized cross correlation finds the best possible alignment between two communication histograms and assess their correlation. Information processing by applications can lead to communication patterns being shifted by τ_{delay}. To overcome the lack of a perfect alignment between two communication networks, we extend the Pearson distance to normalized cross-correlation [BH01]. The normalized cross correlation $\varrho_{r,s}(\tau)$ for two possibly similar communication histograms r and $s \in H$, as defined in Equation D.7, is defined by

$$\varrho_{r,s}(\tau) = \frac{1}{bins} \sum_{t=0}^{bins} \frac{(r_t - \mu_r)(s_{t+\tau} - \mu_s)}{\sigma_r \sigma_s}, \tag{D.11}$$

where μ_r and σ_r are mean and standard deviation of r, such that $-1 \leq \varrho \leq 1$. Anti correlated signals will lead to the result -1 and will lead to us flipping the order of both compared communication histograms. Therefore, comparing two communication histograms will lead to $\varrho_{r,s}(\tau_{delay}) = [0, \ldots, 1]$. The point in time τ_{delay}, where both signals are best aligned can be found by computing

$$\tau_{delay} = \text{argmax}_{\tau \in \{0,\ldots,(t_{max}-t_{min})\} \subseteq \mathbb{N}} \varrho_{r,s}(\tau). \tag{D.12}$$

We regard indirect dependencies $ISDEP$ as events that define a probability space (Ω, F, P_ϱ), where the sample space of all possible indirect dependencies is given by $\Omega = \{\iota_0, \ldots, \iota_n\}$. An indirect dependency event $\iota_i = \{\delta(s_i^j, s_k^l), \delta(s_m^j, s_o^n)\}$ is based on direct dependency events δ. Applying normalized cross correlation according to Equation D.11 to communication histograms, results in a set of events $F \subseteq \Omega$, which are assigned probabilities $\varrho(\tau_{delay}) \in P_\varrho$ ranging between $\varrho(\tau_{delay}) = [0, \ldots, 1]$.

$$p(\iota_{hg}(\delta_h(s_i^j, s_k^l), \delta_g(s_m^j, s_o^n)) \mid \delta_h(s_i^j, s_k^l) \wedge \delta_g(s_m^j, s_o^n)) = \varrho_{r,s}(\tau_{delay}) \tag{D.13}$$

We refer to the events Ω as workflow events and they are the building blocks for the HMM based workflow model derived in Section D.4. An example for this probability space is illustrated in Figure D.2a, which shows an RR dependency and consists of a client d_c sending an HTTP request to a load balancing server d_{lbs}. The load balancing server d_{lbs} then sends an HTTP request to a webserver d^{ws}. The RR dependency, shown in Figure D.2a, can be written as $\iota_{01}\left(\delta_0(s_*^c, s_{80}^{lbs}), \delta_1(s_*^{lbs}, s_{80}^{ws})\right)$. Figure D.2b shows an LR dependency, where the client d_c sends a request to a DNS server d_{DNS}. Afterwards, the client d_c sends a load balancing server d_{lbs}. The LR dependency, shown in Figure D.2b, can be written as $\iota_{23}\left(\delta_2(s_*^c, s_{53}^{DNS}), \delta_3(s_*^c, s_{80}^{lbs})\right)$.

D.4 Workflow Mining

Normalized cross correlation provides an heuristic approach for estimating workflow events and the result is described by a probability space (Ω, F, P_ϱ) described in Section D.3. Based on the probability space, we define the problem of mining workflows as detecting the most likely sequence of hidden states. This is a problem often associated with Hidden Markov Models (HMMs). Using HMMs, it is possible to identify the probability whether a specific workflow is in place or not. We are interested in the most likely sequence of workflows in a given communication data network. The most likely sequence of hidden states can be calculated using the dynamic programming Viterbi algorithm [FJ73].

We understand observed workflow events F as observables drawn from an output alphabet Ω. Normalized cross correlation is a heuristic approach, hence observed workflow events can be untrue and existing indirect dependencies might not be detected. However, repeatedly reoccurring workflow events are very likely to be actual workflow events.

Hence, we understand a workflow event sequence as a sequence of hidden states $\omega \in \Omega$. Every hidden state $\omega \in \Omega$ represents a workflow event and is based on joining to direct dependencies $\delta \in \Delta$. Every workflow $wf_i \in WF$ is modeled as a HMM [RJ86], described by a tuple $\lambda_{wf \in W} = (A, B, \pi, \Omega, F)$, where:

- $A : \Delta \times \Delta \rightarrow [0, 1]$ is a state transition matrix,
- $B : \Omega \times F \rightarrow [0, 1]$ are the observation probability,

- $\pi : \Omega \rightarrow [0, 1]$ is the initial state distribution,
- $\omega = \Delta \times \Delta \times \tau_{delay}$, for $\omega \in \Omega$ is a finite set of states and
- F is a finite set of observations.

Based on tumbling windows $w_{t_i} \in W$

$$W = w_{t_0}, \ldots, w_{t_i}, \ldots, w_{t_{n-1}} \tag{D.14}$$

with a shift Δ_t, network packets are continuously transformed into communication histograms according to Algorithm 2. At the end of every tumbling window, the set of indirect dependencies is derived. Indirect dependencies are considered to be a part of the same workflow WF, if

$$WF = SCC((\Delta, \Omega)), \tag{D.15}$$

where SCC denotes strongly connected components of the graph (Δ, Ω) with the edge set Δ and the node set Ω. This results in a non empty set WF of indirect dependencies, which overlap by at least one direct dependency. The tumbling window described in Equation D.14 allows us to observe workflow sequences $wf_i \in WF$ chronologically sorted over multiple windows. Let us assume that a workflow wf_0 was observed in time window w_{t_0} and a workflow wf_1 was observed in time window w_{t_1}. The workflows wf_0 and wf_1 are merged $wf_0 = wf_0 \cup wf_1$, if $\delta_i \in wf_0$ and $\delta_i \in wf_1$. In other words, two workflows occuring in two different time windows are merged, if they contain the same indirect dependency. Within every workflow the number of occurrences $\eta_{\iota_i \in F}$ for every indirect dependency event is counted.

Based on observed indirect dependencies, the state transition matrix A describes the probability of a state transition from direct dependency $delta_i \in \Delta$ to $\delta_j \in \Delta$ as a matrix element a_{ij}. Dimensions of the matrix are given by the number of network services in a workflow $wf_i \in WF$ and

$$a_{ij} = \frac{\eta_{\iota_{ij} \in F}}{\sum_{k=0}^{n} \eta_{ik}}, \tag{D.16}$$

such that all entries in the same column of state transition matrix A summarize to 1. An example for a state transition matrix A based on the previous example is given in the following.

$$\begin{array}{c} \\ s_*^c \\ s_{53}^{DNS} \\ s_{80}^{lbs} \\ s_*^{lbs} \\ s_{80}^{ws} \end{array} \begin{array}{c} \begin{array}{ccccc} s_*^c & s_{53}^{DNS} & s_{80}^{lbs} & s_*^{lbs} & s_{80}^{ws} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Observed indirect dependencies $\iota \in F$ are assigned a specified alphabet of symbols as observations $F = (\iota_1, \iota_2, \ldots, \iota_M)$. Observations $F \subseteq \Omega$, are a subset of all possible indirect dependencies Ω.

The observation matrix B is derived from the frequency of occurrence of observed indirect dependencies F, such that the probability $B(\Omega|F)$ represents the probability that an indirect dependency $\iota_i \in \Omega$ can be detected by an observation $\iota_j \in F$. Over a period of time $t \in T$, indirect dependencies are monitored and their occurrence patterns over time are reflect by $B(\iota_1, \cdot)$ = [$\eta_{\iota_1|1}$, $\eta_{\iota_1|2}$, $\ldots$, $\eta_{\iota_1|T}$].

$$b_{\iota_i|t} = \frac{\eta_{\iota_i|t}}{\sum_{j=1}^{n} \eta_{\iota_j|t}} \tag{D.17}$$

Such an observation matrix B represents the probability of state transitions over time and can be written as:

$$\begin{array}{c} \\ \iota_1 \\ \iota_2 \\ \vdots \\ \iota_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_T \end{array} \\ \begin{pmatrix} \eta_{\iota_1|1} & \eta_{\iota_1|2} & \cdots & \eta_{\iota_1|T} \\ \eta_{\iota_2|1} & \eta_{\iota_2|2} & \cdots & \eta_{\iota_2|T} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{\iota_n|1} & \eta_{\iota_n|2} & \cdots & \eta_{\iota_n|T} \end{pmatrix} \end{array}$$

The accuracy of the derived workflow model depends on the accuracy of the detected workflow events. Thus, in the following section we will systematically evaluate MONA's performance in detecting workflow events based on network traffic against two other network service dependency analyzers.

D.5 Experimental Evaluation

Goal of the experimental evaluation is to assess, whether MONA is able do deduce a HMM based workflow model by analyzing network traffic. Our network service dependency discovery method identifies workflow events which are building blocks of our workflow model. Thus, the correctness of the HMM based workflow model depends on the correctness of detected workflow events. Workflow events are identified through network service dependency discovery. Hence, our experimental evaluation will consist of a performance comparison of three network service dependency discovery methods. For evaluating the accuracy of detected workflow events, a network traffic trace of a real data communication network is required. This was provided by the disaster recovery site of an energy distribution network, provided an Italian water and energy distribution company. Also, a known ground truth with all existing and non-existing network service dependencies is required. To derive a known ground truth of all network service dependencies in a monitored system, application layer information is required. As multiple third parties are involved in operating a data communication network for energy distribution, including all of them into the derivation of a known ground truth on application layer, would be time consuming and costly. Network simulation allows us to replicate the SCADA network and, additionally, this provides us with a known ground truth. Thus, we rely on network simulation to create synthetic datasets, that are then used estimate precision and recall of the three compared network service dependency methodologies.

D.5.1 Network Simulation

Network simulation is widely used to design and evaluate new protocols and applications. This random network generator can create synthetic data sets with a known ground truth for testing the performance of network service dependency discovery methodologies. The chosen network simulator in the context of this work is ns-3 [AMZ12], and it has been used in numerous domains from simulating peer-to-peer, wireless and ad-hoc networks, business processes or peer-to-peer network. For our experimental evaluation, link, network and transport layers are provided by ns-3. We analyze the characteristics and typical communication patterns of the energy distribution network in order to build a network generator. Network simulation provides multiple benefits compared to real-life data sets. For example, it provides us with a known ground truth including all existing and none-existing network service dependencies.

Additionally, network simulation allows for varying the number of network devices, modifying how frequently they communicate, how many network packets are exchanged and how many network service dependencies exist. Network simulation allows us to systematically add noise to communication patterns and test the robustness to noise of the network service dependency analyzers.

D.5.2 Method for Analyzing the Performance of Network Service Dependency Discovery

To show that MONA surpasses the performance of state of the art network service dependency discovery methodologies, we rely on precision and recall. Network service dependencies consist of indirect dependencies that can be divided into RR and LR dependencies which are introduced in Equation D.8 and D.9. True positives (TP) refer to correctly learned indirect dependencies, false positives (FP) refer to learned indirect dependencies that are false and false negative(FN) are existing indirect dependencies that are not found. Based on these values we can compute precision = $\frac{TP}{TP+FP}$ and recall = $\frac{TP}{TP+FN}$. We use precision and recall to evaluate three network service dependency discover methods: Orion, Sherlock, and MONA.

As introduced in Section D.2, Orion [CZMB08] infers network dependencies between two network services, if the delay between consecutive access follow a constant pattern. As an example for such constant pattern, consider an application which needs consecutive access to two distinct network services. Then, the delay between to dependent network services will follow a non-random distribution. Orion requires a minimum flow count to analyze the delay distribution of network services. Given that enough network flows are provided, Orion calculates a threshold $mean + c \cdot stdev$ over a communication pattern's delay distribution to decide whether to direct dependencies are indirectly dependent. The multiplication factor c is a fixed constant. Sherlock [BCG$^+$07] infers network service dependency based on co-occurrence within network traffic using a dependency interval of 10ms [BCG$^+$07]. As a result, the strength of a network service dependency is computed as the probability of a network service being accessed within a time interval in which another network service is accessed. It focuses on detecting remote-remote dependencies and leverages packet capture running at each end-host to infer dependencies. After inferring network service dependencies, a directed dependency graph is built, modeling network device states as up, troubled or down.

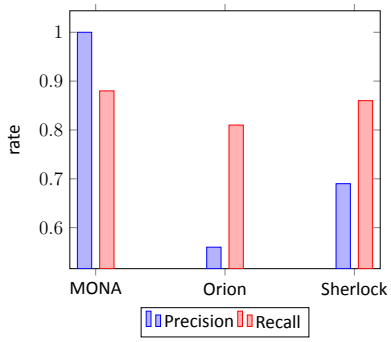
D.5.3 Performance Evaluation

To allow for a comparative evaluation of all network service dependency methods, we abstracted the following characteristics for the energy distribution network:

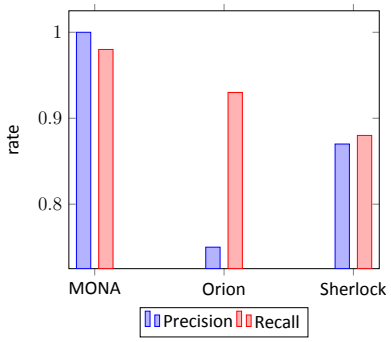
- The more network devices are considered during evaluation, the more network service dependencies exist,
- Network service dependencies between rarely communicating network services consist of 5 to 10 flows per indirect communication and

- Network service dependencies with strongly varying number of flows consist of 5 to 90 flows per indirect communication.

Flow refers to the number of packets exchanged after an initial handshake between indirectly dependent network services. Based on these characteristics, we generate network traffic with the previously described ns-3 based module and assume two different scenarios. Figure



(a) Comparison between MONA, Orion, and Sherlock: 500 devices, 10 direct communication, 70 indirect communication, 5 to 90 flows per indirect communication



(b) Comparison between MONA, Orion, and Sherlock: 200 devices, 10 direct communication, 20 indirect communication, 5 to 10 flows per indirect communication

Figure D.3: Example for identification of indirect dependencies in synthetic networks.

D.3a shows a network with 500 network devices with 10 direct dependencies and 70 indirect dependencies with 5 to 90 flows per indirect dependency. A prior publications [bg15] provided detailed sensitivity analysis of all three methodologies and shows MONA's robustness to threshold variations. We are interested in maintaining a high precision and recall value as we are interested in finding all workflow events and minimizing the false positively identified workflow events. However, Sherlock's and Orion's performance depend on choosing the right threshold, hence, for the performance evaluation, we choose the best possible threshold for all Orion, Sherlock and MONA. The results show that MONA's precision and recall surpasses Orion's and Sherlock's. It should also be noted that recall results for Orion and Sherlock are far better than their respective precision values. Their high recall value means that they tend to find relevant network service dependencies, however their low precision value means that these findings include a lot of false positives. MONA has a higher precision than recall value as we also aim to reduce the detection of incorrect indirect dependencies. MONA's precision and recall value mean that found network service dependencies tend to be true positives, however not all relevant network service dependencies are found.

The second simulated scenario is shown in Figure D.3b shows a network with 200 network devices with 10 direct dependencies and 20 indirect dependencies. Within each indirect dependency, 5 to 10 flows are exchanged. We chose a smaller network size in order to see whether network size influences the performance of the evaluated network service dependency discovery methods. The results of this evaluation mimic our previous findings: with no noise present, MONA has a perfect precision value, however not all network service dependencies are found. With less overall network traffic present, Orion and MONA improve their recall

values. Orion's precision value is unchanged. Surprisingly, Sherlock improves its precision value, but a slight deterioration of its recall value can be seen.

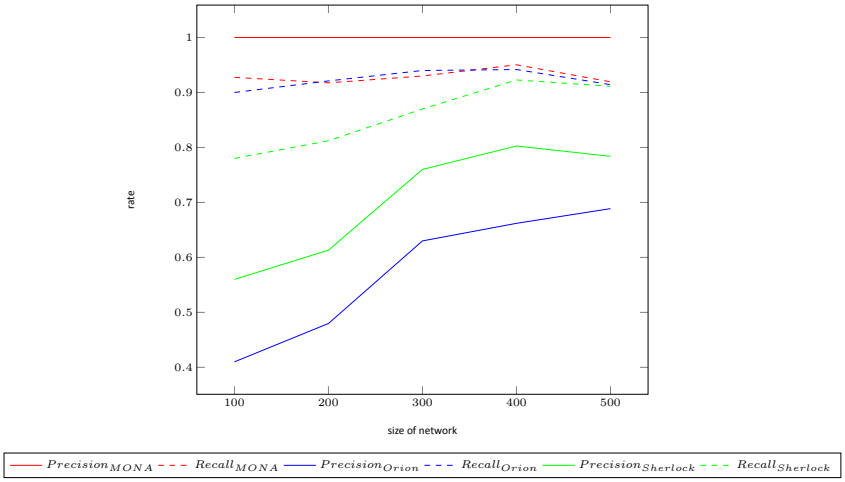


Figure D.4: Average precision and recall for networks of different size.

A comparison between the three network service dependency discovery methods for different sized networks is shown in Figure D.4. We are interested in the average performance of MONA, Orion, and Sherlock for communication networks containing network service of various characteristics. Therefore, we chose five scenarios, each having different amount of direct dependencies and 20 indirect dependencies. This is represented by the x-axis of Figure D.4. For each scenario multiple networks with varying numbers network service dependencies and flows are generated:

- 25 networks with network service dependencies consisting of 5 to 10 flows per indirect dependency,
- 25 networks containing network service dependencies consist of 5 to 50 flows per indirect dependency and
- 25 networks containing network service dependencies consisting of 5 to 90 flows per indirect dependency.

Orion and MONA have a similar recall rate for all five scenarios. Sherlock's recall rate is below Orion's and MONA's recall rate considering networks with few communication. Interestingly, Sherlock's recall rate is similar to Orion's and MONA's recall rate for networks with lots of communication. The precision rate of Orion is below Sherlock's precision rate as we expected from our examples shown in Figure D.4.

As workflow events correspond to indirect dependencies, the correctness of the HMM based workflow model depends on correctly identified indirect dependencies. Correctly identified workflow events, allow deducing a correct HMM based workflow model based on network traffic. This was done based on network traffic provided by the energy distribution network. MONA found all workflows described in the ground truth provided by network administrators. Two currently ongoing workflows were defined in the ground truth and the provided ground

truth consisted of network layer dependencies for each workflow. One of these workflows shows indirect dependencies reoccurring in 1-10 minute intervals, which are caused by SCADA software.

D.6 Conclusion

To evaluate MONA's ability to correctly identify workflow events, the experimental evaluation presented a performance comparison of three network service dependency discovery methods and demonstrates that MONA improves precision and recall rate compared to the state of the art. During first experiments on datasets from the disaster recovery site, MONA often found new network dependencies that had been previously forgotten by network operators. This was generally due to this network relying heavily on third party software that are often also updated and maintained by the third party. Thus, we concluded that deriving manual workflow models is costly and requires specialist know how. Thus, we introduced MONA, a novel workflow mining approach based on network traffic. To the best of our knowledge MONA is the first workflow mining approach, which is able do deduce a HMM based workflow model by analyzing network traffic.

Bibliography

[AMZ12] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical report, NDN, 2012.

[BBB+06] Paramvir Bahl, Paul Barham, Richard Black, Ranveer Chandra, Moises Goldszmidt, Rebecca Isaacs, Srikanth Kandula, Lun Li, John MacCormick, David A Maltz, et al. Discovering dependencies for network management. In *ACM SIGCOMM 5th Workshop on Hot Topics in Networks (Hotnets-V)*, pages 97–102, 2006.

[BCG+07] Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 13–24. ACM, 2007.

[bg15] Cannot be given. This will be given in the final version. 2015.

[BH01] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102. International Society for Optics and Photonics, 2001.

[BPFN08] Tobias Blum, Nicolas Padoy, Hubertus Feußner, and Nassir Navab. Workflow mining for visualization and analysis of surgeries. *International journal of computer assisted radiology and surgery*, pages 379–386, 2008.

[CZMB08] Xu Chen, Ming Zhang, Zhuoqing Morley Mao, and Paramvir Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *OSDI*, volume 8, pages 117–130, 2008.

[FJ73] G David Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[FZMF07] Diogo Ferreira, Marielba Zacarias, Miguel Malheiros, and Pedro Ferreira. Approaching process mining with sequence clustering: Experiments and findings. In *Business Process Management*, pages 360–374. Springer, 2007.

[Her00] Joachim Herbst. A machine learning approach to workflow management. In *Machine Learning: ECML 2000*, pages 183–194. Springer, 2000.

[HK98] Joachim Herbst and Dimitris Karagiannis. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. In *Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop on*, pages 745–752. IEEE, 1998.

[NNL+12] Arun Natarajan, Peng Ning, Yao Liu, Sushil Jajodia, and Steve E Hutchinson. *NSDMiner: Automated discovery of network service dependencies*. IEEE, 2012.

[PM16] V Priyadharshini and A Malathi. Analysis of process mining model for software reliability dataset using HMM. *Indian Journal of Science and Technology*, 9(4), 2016.

[RJ86] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[RVvdA08] Anne Rozinat, Manuela Veloso, and Wil M.P. van der Aalst. Using hidden markov models to evaluate the quality of discovered process models. *Extended Version. BPM Center Report BPM-08-10, BPMcenter. org*, 2008.

[SGVDA08] Minseok Song, Christian W Günther, and Wil MP Van Der Aalst. Trace clustering in process mining. In *Business Process Management Workshops*, pages 109–120. Springer, 2008.

[SZS05] Ricardo Silva, Jiji Zhang, and James G. Shanahan. Probabilistic workflow mining. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 275–284. ACM, 2005.

[TKL+13] Joe Touch, M Kojo, E Lear, A Mankin, K Ono, M Stiemerling, and L Eggert. Service name and transport protocol port number registry. *The Internet Assigned Numbers Authority (IANA)*, 2013.

[VDA04] Wil M.P. Van Der Aalst. Business process management demystified: A tutorial on models, systems and standards for workflow management. In *Lectures on concurrency and Petri nets*, pages 1–65. Springer, 2004.

[VDA11] Wil M.P. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011.

[VDAAdM+11] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *Business process management workshops*, pages 169–194. Springer, 2011.

[VDAvDH+03] Wil MP Van Der Aalst, Boudewijn F van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and Anton JMM Weijters. Workflow mining: a survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

[VDAWM04] Wil M.P. Van Der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142, 2004.

[VDVDA04] Boudewijn F Van Dongen and Wil MP Van Der Aalst. Multi-phase process mining: Building instance graphs. In *Conceptual Modeling–ER 2004*, pages 362–376. Springer, 2004.

E CONTEXT- AND BIAS-FREE PROBABILISTIC MISSION IMPACT ASSESSMENT

Notes *This article is currently under review. This article is an extension of a paper that has been presented at the NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact held in Istanbul, Turkey during June 2015 and at the Workshop on Cyber Defence and Security in Bruxelles, Belgium in September 2015.*

Abstract

Assessing and understanding the impact of scattered and widespread events onto a mission is a pertinacious problem. Current approaches attempting to solve mission impact assessment employ score-based algorithms leading to spurious results. We identify a fourfold problem with score-based algorithms: 1) score-based algorithms enforce deep training of experts to employed frameworks for specification (non-context-free), 2) require reference results for interpreting obtained results (non-bias-free), 3) require assessments outside of an experts' expertise (non-local), and 4) require validation of end -results against ground truth. This paper provides a formal, mathematical model for bias- and context-free mission impact assessment. Based on a probabilistic model we reduce mission impact assessment to a well-understood mathematical problem based on definitions from local expertise and allow for a validation at data level. This is useful for areas and applications where qualitative assessments are required, such as assessments in critical infrastructures or military contexts.

E.1 Introduction

Modeling dependencies of missions on various involved resources is a novel field of research, which pursues the goal of assessing the influences of local impacts on some resources onto a higher goal, i.e., a mission. Assessments somehow require an approach to "spread" locally created impacts onto higher goals, such as missions or processes. Early approaches attempting to solve a problem of mission impact assessment use ad-hoc methods involving newly established algorithms. We argue that such newly created algorithms suffer from multiple discrepancies, which we categorize into four different groups: 1) An expert must first fully understand and be trained in a system before he can assess configurations and parameters. We say, such systems do not provide a context-free assessment. 2) Obtained results from a system require a steep learning curve for interpretation and easily lead to overfitting by a dulling due to learned reference values. This means, results are not bias-free and require knowledge about a system. 3) During configurations, experts are forced outside their expertise, leading to potentially inaccurate specifications. We argue that it is favorable to accept disagreement from multiple knowledge sources instead of enforcing the definition of one allegedly congruent knowledge base. Finally, configurations (compare Problem 1 and 3) were assessed by a possibly overtrained expert and might be inaccurate, but parameters are not verifiable nor can be validated by an independent third party. This means, 4) obtained results from a newly created algorithm must be validated against a ground truth. Ground truths

for occurred events and their exact impact on a mission are often not available in large quantities or are confidential.

To put things into perspective, a non-context-free system requires an expert to understand how an evaluation reacts to a parameter of “5” and how to a parameter of “3”—without the context of the complete framework, such values do not have any meaning and are neither verifiable, validatable, nor are understandable. Further, an end-user becomes biased from interpretation of received results: With an unclear, non-mathematical definition of an end-result, e.g., “yellow”, “3” or “severe”, an end-user intrinsically adapts over time to “normal” results and becomes biased, i.e., a reported “severe” “red” error of category “5” is first taken serious, if it persists for an hour.

In this work, we take a view from different perspectives towards mission impact assessment. We consider three views from three experts from different expertise and combine them inside a well-defined probabilistic graphical model. We provide a context-free assessment of defined parameters and models, which are assessable and can be validated by themselves without knowing their later use. Based on this probabilistic model one finds a well-understood problem: In a complex network multiple events possibly occur, whose local effects must be assessed towards a global effect. Using a probabilistic approach, one can benefit from existing, well-defined and well-understood algorithms to solve this problem and obtain probabilistically sound results that are understandable without the knowledge of our approach. Obtained results are understandable using commonsense and do not suffer from biased interpretations. Furthermore, we present results of two real world use cases on real data using our approach.

The contribution of this article can be summarized as follows: By introducing a well-defined probabilistic graphical model for mission impact assessment, we are able to reduce impact assessment on a well-defined mathematical problem, which allows for a validation of results at data level, which does not require deep training of experts. By resorting to local conditional probability distributions one is able to integrate widespread knowledge from different expertise into one sound model. This is useful for applications, where qualitative assessments are required and perpendicular views from multiple experts onto a problem must be brought inline.

The remaining of this article is structured as follows: In Section E.2 we develop a mathematical model for mission impact modeling based on views from different experts. Based on this model, we discuss mission impact assessment as a formalized problem and its theoretical complexity in Section E.3. We dedicate Section E.4 to a experimental evaluation of scalability, accuracy, and apply our approach to real data in two cyber-security related usecases. Being an emerging field of research, we give an overview on future work in Section E.5 and on related work in Section E.6. We conclude in Section E.7.

E.2 Dependencies and Impacts

In the following, we take a view from different perspectives towards mission impact assessment. We consider three views from three experts from different expertise. We do not enforce an expert to overlook assessments from other experts and expertises (local assessment), and further, do not require that an expert understands or is trained on how his assessment are used inside algorithms and frameworks (context-free assessment). Based on a probabilistic model one finds well-understood probabilistic inference problems that assesses a mission impact from widespread events towards a bias-free result.

Every expert defines a different dependency model, where every modeled entity represents a random variable and a dependency between two entities is represented by a local conditional probability.

Remark 1 (Impact)**.** *We use an abstract term of "impact" in our work in the sense of "not operating as fully intended". The underlying meaning of "intended operation" lies in the usecase of a model.* ▲

E.2.1 Mission Dependency Model (Business View)

In the field of business intelligence, a complete company or organization, i.e., a good one aims to protect, is modeled as a conglomeration of *business processes*. Commonly, business processes are modeled using the business process modeling notation (BPMN) and a business process is modeled as a (dependent) collection of tasks. This modeling approach is well accepted and can be found, e.g., in [dBBdCY13, AJJP13, MTT+10]. Figure E.1 shows a sketch of a BPMN model used throughout this paper.

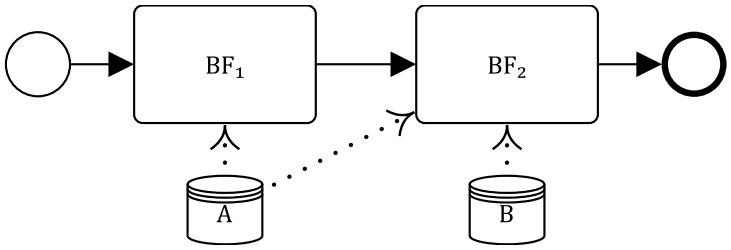


Figure E.1: Example BPMN 2.0 model sketch for the BP_1 business process shown in the dependency model of Figure E.2.

Designing BPMN models is handled manually by an expert from a company or by an external business consultant having a precise expertise in the understanding of business analysis. The business analysis is performed on a pure business perspective and stops at a "resource" level, e.g., it identifies a web-service, but does not describe the dependencies of the webservice on a database or a data center. This is a reasonable approach, as the latter perspective comes from a very different expertise and would require very broad-range experts. Further, an identification of a web-service as a business relevant object is causally more precise in the terms of an operation perspective: A failing database might cause a web-service to not operate as intended and therefore might lead to an unaccomplishment of a business-process. Still, the direct cause for the business-process being unaccomplishable is a rogue web-service and *not* the database. In contrary, an "IT" expert might identify a web-service to be irrelevant, as the crucial point of failure or the point of interest lies in the availability of data from a database. Nevertheless, the latter dependencies must be covered and are discussed in the upcoming subsection.

We extend a model by Jakobson [Jak11] and model mission dependencies as shown in Figure E.2 as a graph of *mission nodes*. We model a *company* as being dependent on its *business processes*. A business process is again dependent on one or more *business functions*. *Business resources* provide business functions. Business resources are part of an infrastructure perspective and—from an operational view—might be irrelevant, but were identified to be business critical. Figure E.2 shows a dependency graph of business relevant objects, based on the preceding presented BPMN model.

As mentioned afore, we represent dependencies in the form of local conditional probability distributions. Every conditional probability describes the probability of failure or *impact* if a dependency fails or is *impacted*. E.g., the probability of the business-function BF_1 (see Figure E.2 and E.1), e.g., "provide access to customer data", failing, given the required business-resource A, e.g., "customer-data-frontend", is 90%. The local semantics of local conditional probabilities allow for an understanding of parameters using common-sense (e.g., "in 9 out of 10 cases, customer data was not accessible for employees during

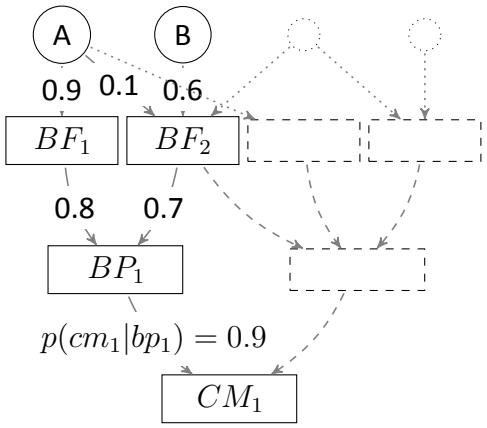


Figure E.2: Mission Dependency Model. Values along edges denote individual conditional probability fragments. Example 3 only uses the solid entities. Consequences and further attributes are omitted in this figure.

frontend-server maintenance") and that the (numerical) assessment can be directly validated by either an expert or through ground-truth.

We consider every mission node as a random variable, where the event $+x$ represents the case that node X is operationally *impacted* and $\neg x$ that is operating as fully intended, i.e., no impact is present.

For ease of parametrization of complete distributions we use commonly known combination functions for easier parametrization of CPDs, such as noisy-or and noisy-and. By the use of combination functions, not complete conditional probability distributions must be parametrized, but solely single fragments of a distribution for every dependency, or edge, in a model.

Definition 6 (From parameters to distributions)**.** *We denote every dependency of random variable Y on X as an* individual *conditional probability fragment $p(+x|+y)$ and $p(+x|\neg y)$. These parameters are* fragments *of a complete conditional probability distribution and are therefore denoted in lowercase. To acquire the local conditional probability distribution $P(X|\vec{Y})$ of node X from* all *its individual parameters $p(X|Y)$ of all dependent nodes $Y \in \vec{Y}$, we employ a non-leaky noisy-or and noisy-and combination functions, as, e.g., described by Henrion [Hen88]. Non-leakiness implies $p(+x|\neg y) = 0$ for every dependency and therefore $P(+x|\neg\vec{y}) = 0$.* ▲

Using Definition 6 the parametrization of a probabilistic graphical model for mission impact assessment becomes easier in certain situations. Notwithstanding, if an expert feels confident to do so, CPDs can be designed directly.

With Definition 6, one obtains a Bayesian network from a mission dependency model, for which one can specify a joint probability distribution over all entities in the dependency model plainly as the product of all local conditional probability distributions.

The example given below shows the intention of using probabilistic dependencies model for mission impact assessments.

Example 3. *Following the rather simple mission dependency model depicted in Figure E.2 (excluding dashed entities), a Bayesian network is evident representing a joint probability distribution $P(CM_1, BP_1, BF_1, BF_2, A, B)$ as*

$$= P(CM_1|BP_1) \cdot P(BP_1|BF_1, BF_2) \cdot P(BF_1|A) \cdot P(BF_2|A, B) \cdot P(A) \cdot P(B)\,, \qquad \text{(E.1)}$$

i.e., the product of all locally defined CPDs. $P(BP_1|BF_1, BF_2)$ *and* $P(BF_2|A, B)$ *are obtained through the noisy-or assumption from* $p(+bp_1|+bf_1)$, $p(+bp_1|+bf_2)$ *and* $p(+bf_2|+a)$, $p(+bf_2|+b)$ *respectively. Due to the absence of global normalization factors in Eq. E.1, locally defined CPDs are interpretable for themselves, e.g.,* $P(BF_2|A, B)$ *and respectively* $p(+bf_2|+a)$ *are understandable* without *a need to consider the degree of, say,* $p(+bp_1|+bf_1)$.

One obtains the probability of impact $+cm_1$ *onto the company* CM_1 *from, say, an observed impact on* $A = +a$ *and none on* $B = \neg b$ *by marginalization from the JPD as*

$$P(+cm_1|+a) = \alpha \cdot \sum_{BP_1} \sum_{BF_1} \sum_{BF_2} P(+cm_1, BP_1, BF_1, BF_2, +a, \neg b) \,, \tag{E.2}$$

with a normalizing factor α, *s.t.* $\sum_{CM_1} P(CM_1|+a) = 1$. *Later, we will define exactly this probability of impact onto a company as a well-defined mission impact assessment. The obtained result* $P(+cm_1|+a)$, *say,* 20% *is a plain conditional probability. Moreover, the obtained result* $P(+cm_1|+a)$ *is defined to be formally correct, given the defined data (defined dependency models) is validated to be correct. This has the advantage that obtained results can be reported to higher authorities without disclosure or explanation of used algorithms or approaches.* ♦

The example shows how a mission impact can be defined based on a probabilistic inference problem. An obtained result of, say, 20% is understandable without a context, i.e., one does not require indepth knowledge of an originating attack or needs to understand how this assessment is obtained. Further, a probability of 20% is interpretable unbiasedly, i.e., every person should come to a similar conclusion on how severe this assessment is.

To detail an effect of an impact further, we define a set of *consequences* for every business process to which a possible failure of the business process might lead. Again, a consequence is modeled as an individual conditional probability stating the probability that a consequence happens, given an impact on the business process. Likewise, one can then calculate the probability that a BP's consequence happens ($+con_{BP}$), given all observed local impacts, say $+a$, plainly as $P(+con_{BP}|+a) = P(+con_{BP}|+bp) \cdot P(+bp|+a)$.

Still, only considering a business view does not cover transitively (or passively) involved resources. To cover distant and widespread local events, which are not directly obvious, we introduce a network dependency model in the upcoming subsection.

E.2.2 Resource Dependency Model (Operation View)

As mentioned afore, an identified critical resources might be threatened *transitively* by further resources inside an infrastructure, or rather, *depend* on further resources. These dependencies are required information, as from a business view, an operationally unimportant (e.g., a frontend server) resource might be identified. From an operation view, e.g., a computation cluster delivering results to the frontend server must be protected. Therefore, a resource dependency model covers dependencies between individual resources, which can be, e.g., individual ICT servers, ICS devices, software components or, in other use cases, manufacturing robots, suppliers, soldiers or vehicles. We follow the same probabilistic approach as before, i.e., every dependency between two resources represents a local conditional probability of impact, if the dependence is impacted, as shown in Figure E.3.

However, in contrary to the mission dependency model, assessing resource dependencies might not be manageable by hand. Complex operation structures render a manual dependency analysis infeasible and

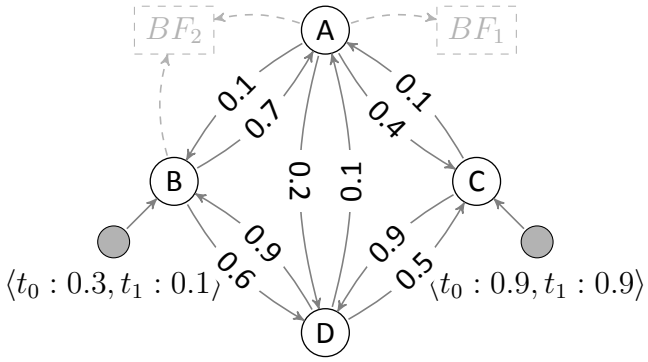


Figure E.3: Resource Dependency Model. Dependencies between B, C would also be possible. Conditional probability fragments are marked along the edges. Grey nodes represent external shock events leading to local impacts. The time-varying conditional probability of local impact given an instantiated external shock event is given below an event's node. Connections to the mission dependency model are sketched in dashed grey.

error prone. Further, dynamically adjusting infrastructures (e.g., as found in IT cloud use cases) make it even unknown to an expert to identify exact dependencies. However, we argue that an expert is able to validate a presented resource dependency model for plausibility. Therefore heuristics are employed based on exchanged information amounts, e.g., traffic analyses in an IT use case, to identify possible resource dependencies. As long as a resource only consumes relevant information for its purpose, every information transfer must motivate some dependency. Moreover, collecting traffic information is a reasonable and feasible effort. Further, under the assumption of *per node* equally distributed entropy and encoding of consumed information, a dependency, i.e., a conditional probability, must be a function of consumed information bits. We, therefore, reduce an infeasible effort for an expert of identifying all dependencies by hand onto finding a heuristic, or rather, validation of a generated dependency model. While a validation of a resource dependency model is expensive, it is a reasonable effort for highly critical infrastructures or operations. The following example demonstrates an approach for an automatic generation of a resource dependency model.

Example 4. *In our use case, we have information about exchanged information at a logical ICT device level covering virtual machines as individual devices. More granular data, e.g., on software layers, was not acquirable. Fortunately, we can assume in our use case that every device drives one purpose. Multiple software applications running on one device will most likely be dependent on each other, and an impact of one software component will very likely lead to an impact of other software components. We therefore say that dependencies at network device level are coarse enough.*

For example, a workstation X consuming different query results from multiple databases will distribute gained and processed information from such queries to other devices. The percentage of received traffic $T_{Y_i,X}$ from every database Y_i towards the total received traffic gives a good guideline for the conditional dependency between them as $p(+x|+y_i) = \frac{T_{Y_i,X}}{\sum_i T_{Y_i,X}}$. In general, this heuristic should apply to all cases where all dependencies of a resource provide similarly encoded data with similar entropy. However, if the workstation further consumes irrelevant 5TB of cat pictures from a local file server, the heuristic will fail, because the workstation also consumed many irrelevant information. Depending on a network or company characteristics other heuristics might be appropriate, e.g., derivation from a mean received amount of data or a mapping onto a σ distribution. ♦

In Section E.4.2 we show that the straightforward approach outlined in Example 4 delivers highly satisfying results in two real world usecases.

Mission dependency models and resource dependency models directly represent an infrastructure and already define a probabilistic graphical model. What is yet missing to perform a mission impact assessment using this model is a source of potential impacts addressed in the following subsection.

E.2.3 Local Impacts (Security View)

A third view involves a security expert able to assess local consequences of events. In the style of reliability analyses using Bayesian approaches we model external shock events inside a network. Every node X might be affected by one or more external shock events $\vec{SE}$, which are prior random variables. An external shock event $SE \in \vec{SE}$ might be present ($+se$) or not be present ($\neg se$), for which a prior random distribution $P(SE)$ is defined. Accordingly, the presence of an external shock event can be known or can be unclear and is assessed probabilistically through its prior random distribution $P(SE)$. We denote the set of observed external shock events (known presence) as a set of instantiations $\vec{se}$ of observed random variables $\vec{SE}_O \subseteq \vec{SE}$. In the case that an external shock event is present ($+se$), there exists a probability of it affecting a node X, expressed as a local conditional probability fragment $p(+x|+se)$. If an external shock event exists and it is not inhibited, we speak of a *local impact* on x. In the case that the external shock event is not present, i.e., $\neg se$, it does not affect random variable X and we write $p(+x|\neg se) = 0$. Every individual conditional probability fragment from an external shock event is treated in the same noisy-or manner as a dependency towards another node, and thus, multiple shock events can affect one node and one shock event can affect multiple nodes.

Classically, a local impact can also be seen as an observation of an impacted node, i.e., $+x$. However, in a Bayesian network approach an observation implies that a reason for the observation is evident from and modeled in the network, i.e., an impact origins from a modeled network and a reason for it lies within an impact of some *other* nodes. By introducing external shock events one gains the ability to model "soft evidence" of local impacts, i.e., one is unsure whether an external shock event might actually lead to a local-impact and affect a node's operational capability. Still, including observations from sensors inside a network (e.g., from an IDS) has significant advances when reasoning about potential sources of impacts and will be further discussed and modeled in Section E.5.

To further detail a local impact assessment, we introduce the concept of temporal aspects to represent that impacts do not necessarily remain constant over time.

Definition 7 (Temporal Aspects)**.** *We define a temporal aspect of an external shock event. We employ the idea of abstract timeslices in which the effect of an external shock event changes. Every abstract time slice then represents a duplicate of the network- and mission dependencies with a different set of local conditional probabilities and prior probabilities of local impacts. We denote time-varying probabilities in a sequence notation as $\langle t_0 : p_0, \ldots, t_T : p_T \rangle$, with $T + 1$ abstract timeslices. In every abstract timeslice i, varying local impacts take their respective conditional or prior probability p_i defined for its time slice t_i.* ▲

Every local impact represents a potential threat and can be, for example, a consequence of a present vulnerability, a countermeasure, a failure or an attack. It lies in the expertise of a security operator to assess a potential *local* impact of those threats. Due to locally viewed CPDs based on combination functions from probabilitiy fragments, an expert does not need to have neither any expertise in resource

dependencies nor an understanding of missions to do so. Further, an assessment of local impact probability can be formally validated through experiments or be grounded on commonsense. The following examples demonstrate how to employ external shock events in an ICT security context and outlines the merits of local assessments.

Example 5 (Response Plan Side Effects)**.** *We employ mission impact assessment to achieve a* qualitative *assessment of potential negative side effects of a proposed response plan to an ongoing or potential attack. We see a response plan as a collection of individual actions affecting a network. E.g., a shutdown of a server might easily reduce the surface of a potential attack. Still, if a critical resource is highly dependent on that server, it might impact a mission even heavier than a potential attack. We consider three mitigation-action types and transform them to external shock events, possibly leading to local impacts.*

The first mitigation action, i.e., an external shock event, is a "shutdown". Obviously, if a node is shut down (+se: the external shock event is present) we can easily say that the probability of local impact, given the shutdown of node X, is 1, i.e., $p(+x|+se) = 1$.

Secondly, employing a patch on a node X might produce collateral damage as well. During installation of the patch, there exists a (low) probability of immediate conflict, e.g., a flat assumption of 10% or a measure published by the software vendor. In a mean time, a patch might enforce a reboot of a network device. This leads to a temporal shutdown and might lead to hardware failure. Finally, after a successful reboot, a replacement of hardware, and/or a restore of a previous backup, the network device will fully resume its operational capability. Using temporal aspects, one is able to model a patching operation in three abstract time slices and define the local impact probabilities of this external shock event to be $p(+x|+se) = \langle t_0 : 0.1, t_1 : 1.0, t_2 : 0.0 \rangle$.

Our third considered mitigation action is the restriction of a connection from node X to node Y, i.e., a new firewall rule. From a technical perspective this operation forbids a transfer of data that might have been crucial for the operational capability of a node Y. Therefore, a firewall rule leads to an operational impact on Y. As a connection between two devices resembles a dependency, one must further actually remove this dependency. Otherwise, one would infer further impacts over a dependency that was prohibited and already assessed locally. To do so, we simply "bend" the forbidden dependency to an observed external shock event +se, s.t., the local conditional failure probability $p(+y|+x)$ becomes a local impact probability $p(+y|+se)$. Another approach, decidable by a security operator, would be to accumulate dropped connections and add an unified local impact for them.

As these external shock events are deliberately placed inside our domain, we model their prior probability to exist as a tautology, i.e., $p(+se) = 1$, and, obviously, fully observe the presence of mitigation actions.

♦

This examples shows how observed events are modeled as external shock events for an assessment. The assessments of *local* impacts are highly beneficial for the example, as not enough, if even any, data is available that allows for an analysis of potential impacts on a company related to executed individual mitigation actions. Without a context-free and bias-free assessment, one needs to generate data for learning and validation of an algorithm: Every mitigation action, and every mitigation action combination, must be executed on all network resources multiple (>1000) times and an impact onto a company must be assessed. Frankly, it is easy to imagine that a company would not exist anymore before experiments have finished. In our approach and usecase, only local assessments of mitigation actions are required

which are validatable locally by using common sense or by using small local experiments, without a need to validate a global assessment (cf. Section E.3).

A second usecase is motivated from an opposite perspective. While response plans discussed in Example 5 are intentionally executed actions on an environment, i.e., local impacts are triggered internally, the following use case considers impacts triggered by external sources, e.g., an adversary.

Example 6 (Vulnerability Impact Assessments)**.** *In a cyber security contexts, vulnerability advisories represent notifications of potential flaws in systems or softwares. It is not always known if a vulnerability is actually "exploitable", meaning, if a flaw can actually be exploited to cause harm to a system. Further, the expected amount of potential harm can be difficult to assess and depend on local configurations of components or further environment constraints. Moreover, inferring if a vulnerability is actually present on a local system requires a deep analysis of local software configurations. These facts motivate to model vulnerabilities as (partially observed) external shock events.*

A vulnerability represents one external shock event SE_V, which affects multiple nodes $\vec{X}$. The prior probability distribution of an external shock event SE_V, i.e., $P(SE_V)$, then represents (i) a probability of existence $P(SE_V^e)$, e.g., it affects all nodes running software Q, but it depends on numerous further uncheckable constraints if vulnerability V actually affects this configuration and represents (ii) the probability of exploitability $P(SE_V^x)$ if a (publicly known) exploit exists for a known vulnerability. $P(SE_V^x)$ is very likely to vary over time for which one can employ abstract time slices. The prior probability distribution $P(SE_V)$ is then obtained by $P(SE_V)_t = P(SE_V^e) \cdot P(SE_V^x)_t$ and can be extracted from CVSS scores based on access complexity, authentication and access vector attributes, where access complexity is likely to vary, i.e., decrease, over time.

Given an exploitable presence of a vulnerability on a node X, a local impact might be created, i.e., $p(+x|+se_v)$. Likewise, this probability represents the expected harm of a successful exploitation of a vulnerability on a node and can vary for every individual node. A required probability fragment for an impact *$p(+x|+se_v)$ is obtainable by a noisy or combination of the CVSS attributes for confidentiality-, integrity- and availability impact.*

All parameters, i.e., $P(SE_V^e)$, $P(SE_V^x)$ and $p(+x|+se_v)$, are qualitatively assessable and understandable for an expert, who can be assisted, or even be replaced, by an automatic extraction from public vulnerability advisories. Generally, abstract time slices could be used to model a vulnerability in different dimensions, e.g., C, I, A. We refrain from this idea and keep the nomenclature of a general impact *on a node.* ♦

Modeling vulnerabilities in a probabilistic model is significantly different from existing approaches to include vulnerability advisories to raise situational awareness, as, e.g., generating attack paths (cf. [JSW02, OGA05]). Attack paths try to address the problem how an attacker might actually compromise the network, i.e., they try to simulate an attacker. In contrary, we intend to raise an amount of situational awareness that provokes a proactive removal of potential impact sources. In fact, examples like StuxNet (see, e.g., [Lan13]) have shown that actual attack paths are only loosely based on an interaction of vulnerabilities, but rather that vulnerabilities represents first stages of attacks. Further, we argue that global effects of local vulnerabilities are not foreseeable by any expert. In our probabilistic model, only local consequences of exploited vulnerabilities must be addressed, and transitive effects are (automatically) assessed due to the resource dependency network. This means, one considers what all could happen *locally* and one does not try to find an actual path of an attack or somehow assess global effects at once. An actual use case demonstration for this example is given in Section E.4 and we show that one

obtains an assessment that is understandable more context-free, namely "there exists a x% probability of compromise to our company" instead of "there exists an attack path over cve-xy, cve-zt, cve-ix, cve-po", which is only understandable to an it security expert. The mathematical foundations for a context- and bias-free probabilistic mission impact assessment from the defined static dependency models are introduced in the following sections.

E.3 Probabilistic Mission Impact Assessment

One obtains a probabilistic graphical model from a mission dependency model and a resource dependency model. Informally speaking, in the resource dependency model, some nodes are threatened by external shock events, and, as nodes are dependent, a threatened node might again threaten another node, leading to a "spread" of impacts. We say, a node is threatened by an external shock event *transitively*. In the end, there exists a probability that even a business process or the complete modeled company (mission) is threatened transitively by various external shock events. To recall, to be threatened by an external shock event (might) lead to an impact; and it is a well-defined problem of calculating this "might"-probability of being impacted due to an external shock event, which is what we call the mission impact assessment as done in Example 3.

Definition 8 (Mission Impact Assessment, MIA)**.** *A mission impact assessment of a mission node MN is defined as the conditional probability of MN being impacted ($+mn$), given all observed external shock events $se \in \vec{se}$, i.e., $P(+mn|\vec{se})$, where the effects of local impacts due to $\vec{se}$ are mapped globally based on mission-dependency and resource-dependency graphs. Note that $\vec{se}$ includes present ($+se$) and absent ($\neg se$) shock events and that some shock events are unobserved. The task of obtaining $P(+mn|\vec{se})$ is defined as the MIA problem.* ▲

Given Definition 8 it is the task of an mission impact assessment to solve the MIA problem, i.e., to obtain the probability $P(+mn|\vec{se})$. From a probabilistic point of view, a sound definition of an overall joint probability distribution as demonstrated in Example 3 is required. As demonstrated, this is given for the mission dependency graph, because it is a directed acyclic graph and represents a Bayesian network. However, in the resource dependency graph an acyclicity constraint cannot be assumed and Bayesian network semantics are not well-defined for cyclic graphs. One could see a network dependency graph as a Markov random network, which, however, due to a needed global normalization factor, destroys an intended local view on probabilities. Due to the employed noisy-or assumption, one can see the probabilistic model as a probabilistic logic program, where every "path" $w_i^{MN} \in \vec{w}^{MN}$ from an external shock event $SE \in \vec{SE}$ to the mission node MN is a conjunction of Boolean random variables and is a sufficient *proof* for satisfying $\{MN = true\} = +mn$. Due to the noisy or assumptions, $\vec{w}^{MN}$ then represents a disjunction of conjunctions. Every proof w_i^{MN} exists with a probability $P(w_i^{MN})$, where $P(w_i^{MN})$ is the product of all probabilities in this proof. Let $\mathbf{P}(w_i^{MN})$ denote the probability viewed as a set. $P(+mn|\vec{se})$ is then the probability that at least one proof holds, or rather, the probability that the disjunction of conjunctions is satisfied, i.e.

$$P(+mn|\vec{se}) = \bigcup_i \mathbf{P}(w_i^{MN}) = P(\vec{w}^{MN}) = P(\{\bigvee_i w_i^{MN}\}) \,, \tag{E.3}$$

where not all $\mathbf{P}(w_i^{MN})$ are disjoint (see Figure E.4) and it is worth noting that proofs share common "edges" and end in common shock events. Calculating $\bigcup_i \mathbf{P}(w_i^{MN})$, i.e., an exact solution to the MIA problem, is also known as the probabilistic satisfaction problem and is also used in the Problog reasoning

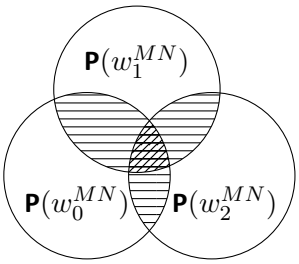


Figure E.4: Illustration of $\mathbf{P}(w_i^{MN})$ viewed as sets. Overlapping parts (filled with patterns) are commonly shared probabilities between proofs and are not allowed to be counted twice (or even multiple times) when calculating $\bigcup_i \mathbf{P}(w_i^{MN})$.

framework [RKT07]. Note that, as every edge represents a conditional probability and a shock event is a prior random variable, plain summation would double count these probabilities and lead to spurious results. This is exactly the issue from which many fudge-factor based "propagation" algorithms in ad-hoc solutions suffer.

The defined probabilistic mission impact assessment $P(+mn|\vec{se})$ directly originates from the definition of all defined dependency-models and represents an inference problem in a probabilistic graphical model. Therefore, if locally defined dependency-models are validated to be correct, an obtained impact assessment $P(+mn|\vec{se})$ is *validated, too*. This means that only locally defined parameters and *not* a complete mission impact assessment must be validated by an expert or against ground truth. Moreover, as $P(+mn|\vec{se})$ is a conditional probability, it is bias- and context- free understandable and is therefore more suitable for reporting along a chain of command.

An exact solution to the MIA problem by a calculation of $\bigcup_i \mathbf{P}(w_i^{MN})$ is possible by the inclusion and exclusion principle and the Sylvester-Poincaré equality. Still, calculation is exponential in the number of proofs due to the subtraction of all overlapping sets and is therefore not practical. We therefore approximate a solution by the use of a Monte-Carlo simulation.

E.3.1 Monte-Carlo Approximation

To find an approximate solution to the MIA problem, we use a two step approximation procedure.

For every mission node MN, there exists a Boolean formula $\vec{w}^{MN}$ as a disjunction of conjunction over Boolean random variables $\vec{B}$. However, Boolean random variables in $\vec{B}$ take their respective truth value according to a probability distribution. As every conjunction in a disjunction is sufficient to proof $+mn$, we speak of them as probabilistic proofs.

Definition 9 (Probabilistic Proofs)**.** *For every business resources $BR_i \in \vec{BR}$ let $\vec{w}^{BR_i}$ denote the set of all proofs and let $w_j^{BR_i}$ denote the j^{th} proof. Let $\vec{w}$ denote the super-set of all found proofs. Every proof $w_j^{BR_i}$ is a set of individual conditional probability fragments $p(+x|+y)$, representing an edge, i.e., a dependency of X on Y. The product of all probability fragments $p(x|y) \in w_j^{BR_i}$ is the satisfaction-probability of a proof $P(w_j^{BR_i})$. Every proof $w_k^{BR_i}$ for which holds $\exists j : w_j^{BR_i} \subseteq w_k^{BR_i}$ is irrelevant for calculation and $\vec{w}$ is a finite set. Informally this means, during proof search along one "path" an already visited node must not be visited again and we cannot get stuck in infinite loops.* ▲

Based on a set of probabilistic proofs, one obtains an algorithm to find an approximate solution to the MIA problem for all nodes of a mission dependency model.

Definition 10 (Algorithm to the MIA Problem)**.** *Let M be a mission dependency model consisting of business resources $\vec{BR}$, business functions $\vec{BF}$, business processes $\vec{BP}$ and a business company BC. Let R be a resource dependency model. Let $\vec{SE}$ be a set of external shock events affecting resources in R and let $\vec{se}$ be a set of observed shock events.*

Then, for every business resource $BR_i \in \vec{BR}$, $\vec{w}^{BR_i}$ is obtained by a depth-limited search from BR_i through R to any $SE \in \vec{SE}$. A complete sample $\vec{s}$ of all random variables in $\vec{w}$, M and $\vec{SE}$ is drawn according to each respective random distribution. Consecutively, for every BR_i the satisfaction of $\bigvee \vec{w}^{BR_i}$ by $\vec{s}$ is checked and marked on BR_i by a respective $+br_i$ or $\neg br_i$. Subsequently, for every $BF_i \in \vec{BF}$ the satisfaction of all $\bigvee_{BR_i} p(+bf_i|BR_i) \wedge BR_i$ is checked and marked accordingly. Respectively for $\vec{BP}$ on $\vec{BF}$ and BC on $\vec{BP}$. Every satisfaction of a mission node $MN \in M$ is counted by hit_{MN}. Sampling and checking is iterated n_S times. ▲

Theorem 1 (Solution to the MIA Problem by Algorithm of Definition 10)**.** The algorithm described in Definition 10 approximates a solution to the MIA problem on every mission node $MN \in M$ using n_S iterations by $P(+mn|\vec{se}) = \frac{hit_{MN}}{n_S}$. For a depth-limited search, the algorithm scales linear with the number of edges in R, $|\vec{SE}|$, $|\vec{w}|$ and n_S. For an infinite number of samples n_S and an unlimited depth-search, the algorithm generates an exact solution to the MIA problem. ▲

A proof is given and empirically evaluated in the following section, after a short demonstration of the approximation procedure.

Example 7. *Consider Figure E.3, where an identified mission critical resource A (compare Figure E.2) is threatened (transitively) by local impacts on nodes B and C. Say both external shock events are observed to be present, i.e., $\vec{se} = \langle +se_B, +se_C \rangle$. We exclude the dependency of BF_2 on B and temporal aspects for brevity. Through depth-first search one finds proofs $\vec{w}^A$ as*

$$\begin{aligned} w_0^A &= \{p(+a|+b), p(+b|+se_B)\} \\ w_1^A &= \{p(+a|+b), p(+b|+d), p(+d|+c)p(+c|+se_C)\} \\ w_2^A &= \{p(+a|+c), p(+c|+se_C)\} \\ w_3^A &= \{p(+a|+c), p(+c|+d), p(+d|+b), p(+b|+se_B)\} \end{aligned} \tag{E.4}$$

Additional proofs, e.g., $w_\emptyset^A = \{p(+a|+b), p(+b|+c), p(+c|+b), p(+b|+se_B)\}$, are redundant, as, here, w_0^A is always (already) satisfied, if $w_\emptyset^A$ is satisfied. After finding these proofs, finding proofs to higher nodes in a mission dependency model, say, to BF_1, is trivial, by simply appending $p(+bf_1|+a)$ to every proof of A. Subsequently, the same holds for BP_1 and CM_1.

For simulation, at first every used random variable is sampled. Let $\vec{RV}$ be the vector of all random variables included in all proofs, i.e., $\vec{RV} = \langle p(+a|+b), p(+b|+se_B), p(+b|+d), p(+d|+c), p(+c|+se_C), p(+a|+c), p(+c|+d), p(+d|+b), p(+bf_1|+a), p(+bf_2|+a), p(+bp_1|+bf_1), p(+bp_2|+bf_2), p(+cm_1|+bp_1)\rangle$. Let $\vec{s}$ denotes a sample of $\vec{RV}$, e.g., $\vec{rv} = \langle +, +, +, +, +, \neg, \neg, \neg, +, +, +, +, + \rangle$, where $+$ represents a true sample, and $\neg$ a false sample.

Subsequently, for every identified critical resource, i.e. A, we check if at least one of its proof is satisfied, i.e. if $\bigvee \vec{w}^A$ is satisfied. We obtain that w_0^A is satisfied and this satisfies A. The circumstance that w_1^A is also satisfied, but w_2^A and w_3^A are not satisfied is irrelevant and further checks can be skipped. Subsequently, we can check the remaining mission dependency graph for further satisfactions in this sampling round. As A and $p(+bf_1|+a)$ are satisfied, BF_1 is satisfied (marked by $+bf_1$) as well. The same

holds for BF_2. Likewise, BP_1 is satisfied as well as CM_1. Every satisfaction is marked as a successful Monte-Carlo round and increments a mission node's MN hit counter hit_{MN}.

This procedure is repeated n_S times, i.e. $\vec{rv}$ is sampled and $\vec{w}$ is checked. Finally, every operational impact assessment of a mission node MN, represented by the conditional probability $P(+mn|+se_C, +se_B)$, is approximated by $P(+mn|+se_C, +se_B) = \frac{hit_{MN}}{n_S}$. ♦

For implementation, some remarks are made on optimizing the procedure.

Remark 2 (Proof Check)**.** *Checking all proofs during one Monte-Carlo round is highly optimizable. $\vec{w}^{BR_i}$ can be sorted descending by $P(w_j^{BR_i})$, s.t. most likely holding proofs are checked first and subsequent checks can be skipped once a satisfied proof is found. Further, a proof $w_j^{BR_i}$ can be sorted ascending by its individual local conditional probability fragments, s.t. most unlikely random variables are checked first and further checks inside one proof can be skipped. Further, a proof w with $P(w) < \frac{1}{n_S}$ will statistically never be drawn, i.e., all such proof can be ignored during simulation and check.* ▲

Remark 3 (Temporal Aspects Implementation)**.** *We introduced that an external shock event, can have different conditional local probabilities depending on an abstract time slice. This means we have a varying probability inside one proof $w_j^{BR_i}$. Naively, one could perform a Monte-Carlo simulation for every abstract time slice through complete duplication. However, this would redundantly simulate all non-varying probabilities. We therefore partition $w_j^{BR_i}$ in a non-varying set of conditional probabilities, i.e. a "path" to an impacted node, and a set of varying conditional probabilities, i.e., a set of local impacts.* ▲

E.4 Complexity and Experimental Evaluation

As a central theme, we focused on actual feasibility of our proposal and we demonstrate that our approach scales well, i.e. linear, with a graph's complexity. In the following, we give a short expected summary of the complexity of our approach and evaluate it experimentally. We provide an experimental validation of expected time-complexities, as well as an demonstration of our approach applied to real data for Example 6.

E.4.1 Complexity Analysis and Experiments

Evaluation and demonstration of the computational complexity of our presented approach is difficult, as it depends on the graph structure of the network and the processed response plan. We therefore use random graphs containing n_N nodes and $n_E = n_N^2 \cdot 0.1$ edges while assuring that every node is at least bidirected. By doing so one obtains a fully connected graph with, approximately, a 10% chance of two nodes being directly connected. In every experiment, we process $n_{SE} = n_N \cdot 0.1$ randomly placed external shock events, i.e., 10% of all nodes are possibly impacted. We measure the time t_{ps} required for finding all n_P proofs up to depth d_{max}, and t_{sim} required for simulating all found proofs n_S times. Every experiment is repeated in 50 different random graphs.

Complexity is differentiated between both steps of the approximation procedure. Given a constant maximum search depth d_{max}, depth-limited search (DLS) scales linearly with the number of edges n_E, as also experimentally evaluated in Figure E.5. Further, DLS scales slightly with the number of placed local impacts n_{SE} (compare Figure E.9), as a pre-computation of shortest distances to local impacts per node can eliminate dead ends early. We write, the time required for a proof search is a function proportional to $t_{ps} = f(n_E, d_{max}, n_{SE})$.

Remark 4 (DLS). *DLS scales exponentially with a specified maximum depth d_{max}. In general and for our example, the maximum depth should be chosen in the range of the average path length inside a given graph, s.t., almost every node is considered at least once. In order to better scale with maximum depth it is reasonable to allow a rational d_{max}, where a depth $0 < d_{dec} < 1$ resorts to the best d_{dec}, i.e. most dependent, children.* ▲

Monte-Carlo simulation, i.e. sampling and checking of proofs, scales linearly with the number of found proofs n_P (compare Figure E.6) and the number of Monte-Carlo samples n_S per business resource (compare Figure E.7), i.e., $t_{sim} = f(n_P, n_S)$. Naturally, the number of proofs n_P scales with the number of local impacts n_{MA}, of edges n_E and the maximum proof length d_{max} (compare Figure E.8), i.e., $n_P = f(n_{SE}, n_E, d_{max})$. In summary, all experimental results match expected theoretically complexities.

In order to verify the correctness and convergence of the proposed Monte Carlo approximation procedure (see Section E.3.1), Figure E.11 and E.10 show the absolute error of the proposed Monte Carlo approximation towards an exact solution based on the inclusion- and exclusion principle. Experiments where evaluated on 100 different networks and convergence results where considered on up to $n = 5\,000\,000$ samples. A mean of an absolute error $\bar{E}$ of the proposed approximation follows $\bar{E}(n_S) = 0.2\sqrt{n_S}^{-1}$ for n_S samples and corresponds to an analytically derived error behavior (cf. [Owe13, Sec. 2.2]), which is expected to be proportional to $\sqrt{n_S}^{-1}$. The variance of the absolute error σ_E for n_S samples follows $\sigma_E(n_S) = 0.17\sqrt{n_S}^{-1}$.

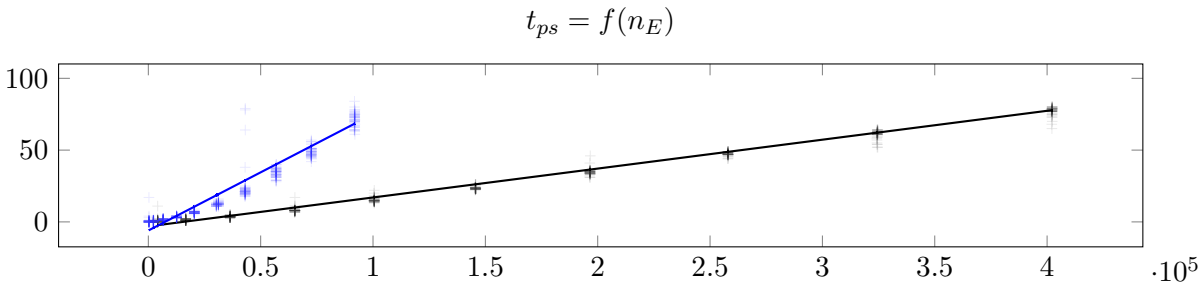


Figure E.5: Proofsearch time (t_{ps} in ms, ordinate) is linear with the number of edges in the graph (n_e, abscissa). Black: $d_{max} = 3$, Blue: $d_{max} = 4$. n_N is linearly increased, meaning a quadratic increase of edges. $t_{ps} = f(n_E, d_{max}, n_{MA})$, n_{MA} only very slightly.

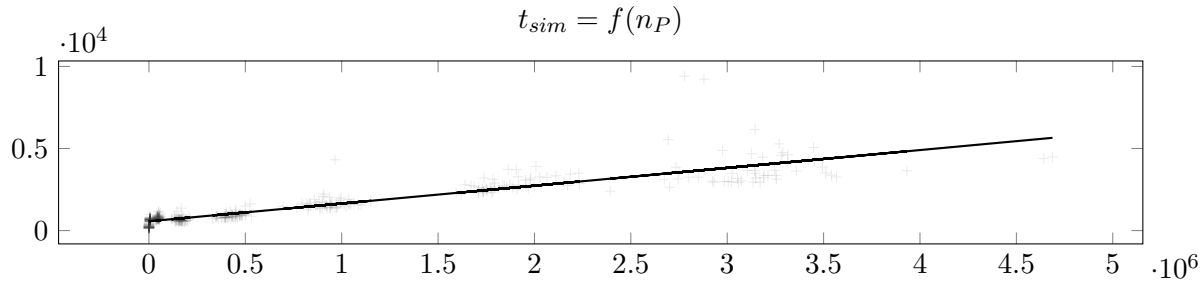


Figure E.6: Simulation time (t_{sim} in ms, ordinate) is linear with the found proofs (abscissa). $n_S = 10000$, $d_{max} = 5$. n_N is linearly increased, meaning a linear increase of shock events and a quadratic increase of edges, both increasing the amount of proofs. Linear time complexity is also achieved for very large proof sets.

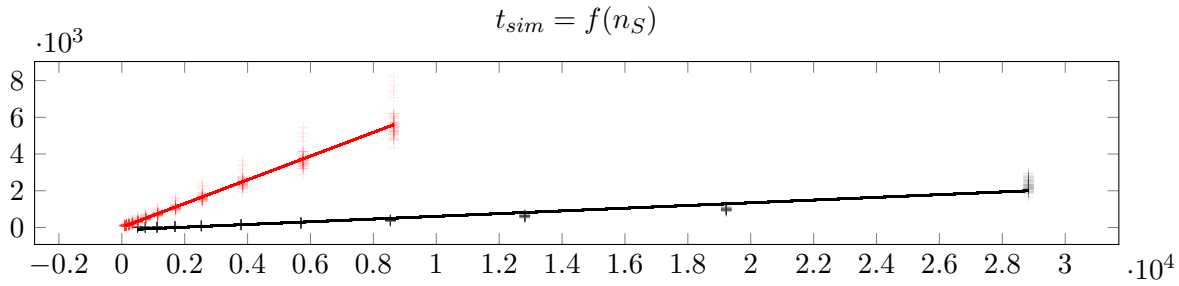


Figure E.7: Simulation time (t_{sim} in ms, ordinate) directly correlates with the number of Monte Carlo simulations (abscissa). Unsimulatable proof cut-off (black) increases simulation time drastically (red without cut-off). $n_N = 500$, $n_{SE} = 50$, $d_{max} = 4$. n_S is increased exponentially and every measurement is repeated for 50 different random graphs.

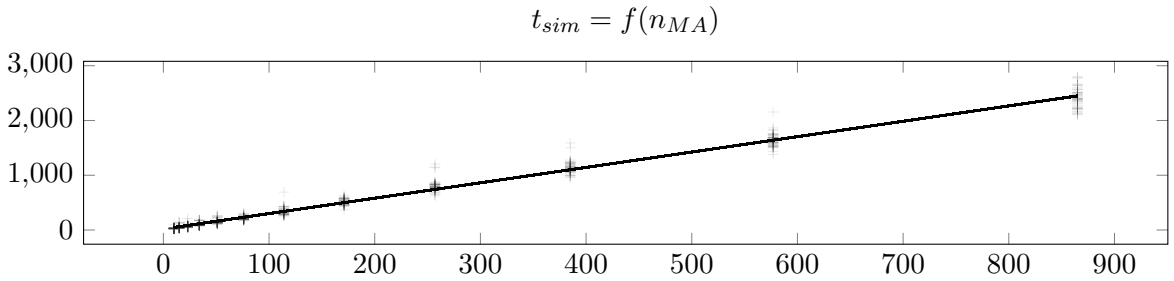


Figure E.8: Simulation time (t_{sim} in ms, ordinate) directly correlates with the number of shock events (abscissa). Constant $n_N = 1000$, $d_{max} = 3$, $n_S = 1000$. n_{SE} is increased exponentially and repeated in 50 different random graphs.

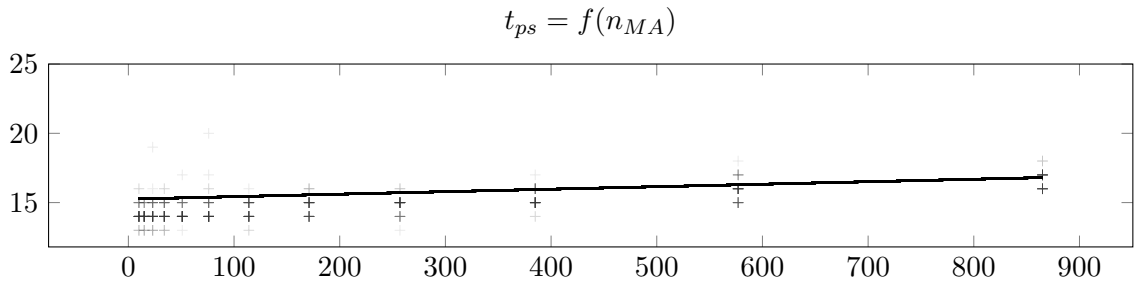


Figure E.9: Proofsearch time (t_{ps} in ms, ordinate) is negligible dependent of the number of shock events (abscissa). t_{ps} correlates with the number of edges in the graph. Measurements collected during Figure E.8.

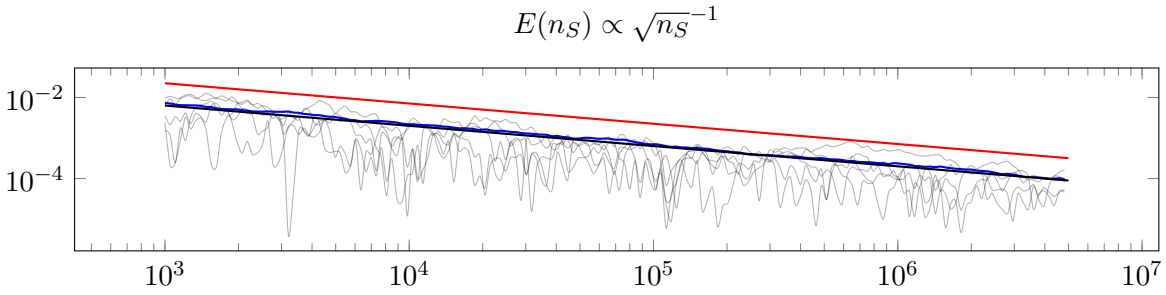


Figure E.10: Absolute error (ordinate) of Monte Carlo approximation compared to an exact solution for large sampling domains (number of Monte Carlo samples, ordinate). Evaluated for one constant network, 100 times. Even for large sampling spaces the mean absolute error (blue) follows $\bar{E}(n_S) \approx 0.2\sqrt{n_S}^{-1}$ (black). Upper three-sigma bound displayed in red and five sample evaluations given for reference in gray. *Double logarithmic plot.*

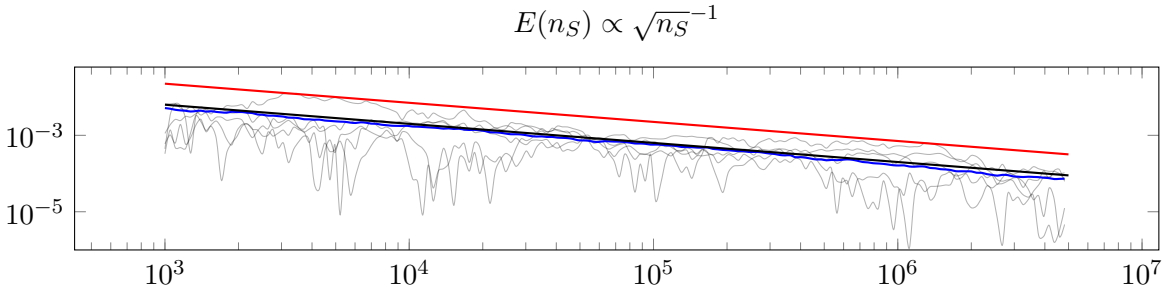


Figure E.11: Absolute error (ordinate) of Monte Carlo approximation compared to an exact solution for large sampling domains (number of Monte Carlo samples, ordinate). Same evaluation as in Figure E.11, but for 100 different, randomly generated problems. *Double logarithmic plot.*

E.4.2 Usecase Experiments

In Examples 5 and 6 we discuss two use cases for the introduced probabilistic mission impact assessment, and give an example for obtaining a resource dependency model automatically in Example 4. In the following to paragraphs, we apply all examples to two real world use cases and demonstrate that the approach is directly applicable, delivers satisfying results and is greatly accepted by experts.

SMIA Challenge In 2011 Sommestad [SH13] conducted an experiment at the information warfare lab of the Swedish defense research agency which gives us the opportunity to demonstrate Example 6 for vulnerability impact assessment. In the 2011 experiment, codenamed SMIA2011, a complete network consisting of multiple domains was set up containing multiple ICT servers, clients, mailservers, firewalls, ftps, webservers, even SCADA server, etc. User behavior was simulated inside each domain by action scripts, e.g., checking webservices, emails and downloading files. Intrusion detection systems provided information to one team in charge of monitoring the complete network and noting suspicious behavior. Another team was in charge of infiltrating the network. Both teams carefully documented their approaches and all network traffic was recorded over six days.

In summary and most noteworthy, the attacking team was able to ex-filtrate all mail messages and change parameters of a SCADA server. For our analysis we consider these servers as mission critical

resources. While some attacks built on each other, the most impacting attacks were almost uncorrelated. Further, vulnerabilities actually played an insignificant role during attacks. As our approach does not build up on actual attack sequences, but rather considers vulnerabilities as points of interest, we evaluate if our approach is able to raise a significantly high enough situational awareness to be concerned about a compromise, i.e., impact, on the identified mission critical systems.

In more detail, the attacking team firstly discovered a misconfigured service running on one mailserver m_1 that allowed for an extraction of an (encrypted and later decrypted) password file. Decryption of the password file from m_1 allowed for a privileged ssh connection on m_1 that, further, allowed for tunneled attacks to two hosts f_6 and o_6 that were not reachable previously. Exploitation of a same known vulnerability on f_6 and o_6 gave a remote shell that revealed further user-passwords. The extracted passwords allowed for downloading all mailboxes from all domains, most likely due to reused passwords in multiple domains. Another vulnerability was exploited directly on another host h_a. While revealing new passwords, no further attacks build up on it. Nevertheless, it could have been an excellent starting point for the following and most interestingly attack: The attacking team was able to completely manipulate all employed firewalls and allowed itself complete access to any node on any domain. Firewalls were only secured by a simple password ("password"), but it could have been extracted in one of the previous exploits. Due to the broadened reachability, the attackers had free access to a (otherwise completely unsecured) SCADA server, on which they successfully changed various parameters. Note that the last attack not obtained through any vulnerability and therefore would not have been detected by any analysis of vulnerabilities alone. In fact, no vulnerability was found, nor exploited on any mission critical resource.

To obtain a resource dependency model, we follow Example 4, but only use recorded traffic of the first day prior to any severe actions carried out by the attacking team (such as flooding the IDS and changing firewall configurations). An obtained dependency model as shown in Figure E.12 seemed plausible, but dependency degrees showed up to be imbalanced: The amount of analyzed traffic of the first day was short and user scripts did not generated a realistic amount of traffic. Further, no operator was simulated to control or monitor the SCADA server, which is why it did not appear in the dependency model analysis. To overcome these circumstances, a minimal dependency probability of 5% is assumed and the SCADA server is manually modeled to be dependent on an operator from the same domain and vice versa. Note that these manual corrections are exactly foreseen in our dependency model. A domain expert is assisted by a heuristic delivering a locally interpretable model, which is, if needed, corrected and consecutively validated in his expertise.

External shock events are modeled as described in Example 6. Attackers exploited three different vulnerabilities on four hosts (shown in red in Figure E.12). Three hosts are part of the domain containing m_1 and one in the domain of scada server f_6. For all vulnerabilities, exploits are publicly known and integrated into various frameworks, as, e.g., metasploit, which is why we assume $P(\text{+}se_V) = 1$ for all of them. Local impact probabilities are adapted frankly according to their respective CVSS score divided by 10.

Based on the defined local impacts, resource dependency model and mission dependency model we obtain that there exists a probability of 23.4% of impact, i.e., compromise, of the firstly attacked mail server m_1 and that there exists a 7.8% probability of compromise of the discussed SCADA server f_6. Both servers were in fact compromised, but through ways unforeseeable from any vulnerability-focused analysis. We argue that a probability in these ranges should not be ignorable by any person confronted with these results. Even the other compromised mail servers from domains without present shock

events, featured probabilities of about 8.1%, 8.1%, 10% and 7.6%.

Considering all critical resources to be equally part of one business process of a, say, cloud company, an impact probability of 38.3% on the company's mission is assessed.

We admit that our assessment might be overcritical and that impact probabilities might be significantly over-assessed. Nevertheless, this approach is completely transparent and understandable throughout. Every defined parameter can be grounded on expert assessments or historical evaluations and, finally, a produced assessment of "Due to a set of local, widespread impacts, there exists a probability of 38.3% that our mission will be compromised" is understandable and not dismissible. We argue that this assessment is eligible for a reporting throughout a command-chain and understandable by every instance. In contrary, a non-bias- and non-context-free assessment of, exaggeratedly said, "vulnerabilities lead to a mission impact of 25764.324" does only make sense for a deeply trained expert or given reference results and is not suitable for any reports.

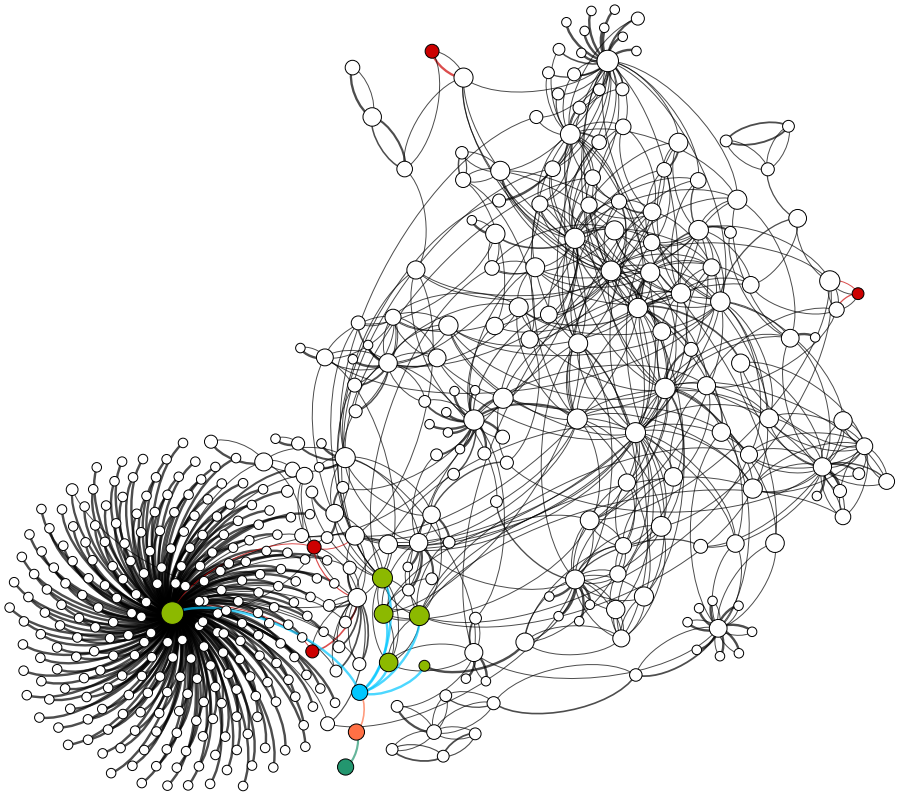


Figure E.12: Resource dependency model extracted from one day traffic captures for the SMIA challenge use case. Red nodes are directly impacted by vulnerabilities and affect other nodes transitively (first-step edges highlighted in red) and are remotely placed from mission critical devices (green). Thicker and darker edges represent higher dependency degrees. For visualization some insignificant dependencies are removed. In fact, the generated network is far from a fully meshed network (only 1% of all possible edges are extracted.) $n_N = 475$, $n_E = 2431$. Visualized using Gephi [BHJ09].

ACEA Use Case The PANOPTESEC integrated research project aims to "deliver a beyond-state-of-the-art prototype of a cyber defencse decision support system" [Pan13]. As part of this project, we employ the derived probabilistic mission impact assessment for both use cases as outlined in Example 5 on response plan assessments and Example 6 for a static vulnerability assessment. We are able to apply and test the complete approach with the PANOPTESEC's use case partner, ACEA SpA, Italy's largest water services operator and one of the largest energy distribution companies in Italy [Ace16]. To deploy a probabilistic mission impact assessment, all three models need to be created. In a team session of two business experts from the company and one IT specialist, a complete mission dependency model was creatable in less than three hours. Admittedly, business experts showed to be reluctant to give assessments of CPDs, but intrinsically where able to understand all parameters. Given a set of choices, experts quickly agreed on an assessment and validated a complete mission dependency model, which is displayed in Figure E.13 in combination with the resource dependency model.

A resource dependency model, depicted in Figure E.13 was automatically extracted from recorded traffic in a redundant, backup environment with simulated behavior of SCADA devices as outlined in the Example 4. The extracted model was validated by an external IT specialist consultant to the company to be reasonable. However, we admit that not all individual probability assessments where validated, but critical dependencies where validated to be included and to bear a reasonable dependency degree. In the use case, a large amount of remote SCADA devices are present. Requiring a remote connection, dependencies on routing equipments are critical in this domain. In order to cover these resources as well, traffic is not solely analyzed on a logical level, but also on a physical level. To be precise, every connection is established between two logical devices, e.g., identified by two IPs, through two physical devices, e.g., identified by two MAC addresses. Through the use of a global inventory of all IT related resources, IPs and MACs are mapped towards unique identifier and a dependence of each resource on its communication-establishing device is added. For example, say, a traffic recording includes a connection from MAC_1, IP_1 to MAC_2, IP_2. Say IP_1 maps to ID_1, IP_2 to ID_2 and MAC_1 to ID_3, MAC_2 to ID_4; then a dependency of ID_2 on ID_1 is added, as well as a dependence of ID_1 on ID_3 and of ID_2 on ID_4. By doing so, one considers that a potential impact on a router, directly affects all resources communicating over said router.

In addition to the vulnerability assessment, as outlined in the previous use case and Example 6, in this use case the response plan assessment plays a major role. Assessing how response plans affect an environment is a completely novel problem and no large datasets are available and we must rely on a validation by an expert. In fact, the IT specialist and product owner of the PANOPTESEC system validated the impact assessments outlined in Example 5. As part of the PANOPTESEC system, response plans are proposed automatically by different components in different situations. However, response plans are proposed based on their effectiveness against attacks and financial benefits, which are both necessarily in line with an assessment of operational impact. For example, a shutdown of highly critical node will certainly eliminate all attacks targeted towards that node and is financially highly traffic, as this response plan does not involve almost any cost. However, this response is catastrophic when considering implications on the mission, i.e., company.

E.5 Extensions to Dynamic Mission Impact Assessment

Temporal aspects introduced in Definition 7 introduce a need for mission impact assessments over *time*. Further, in rapidly changing environments, i.e., dependencies of resources change rapidly over time, require that a mission model is able to be evaluated over time. An extension of Bayesian networks

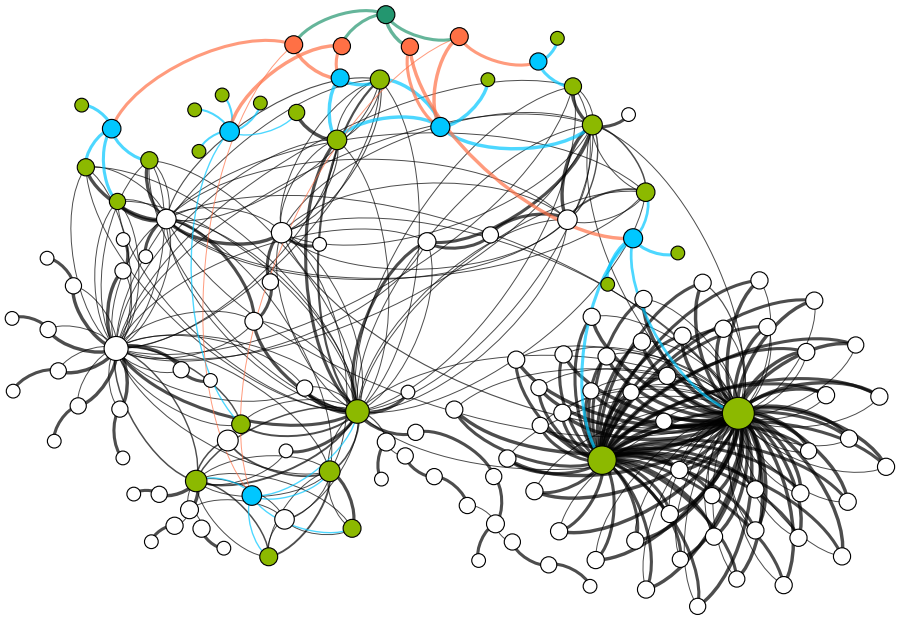


Figure E.13: Resource dependency model extracted from one day traffic captures of a use case partner resembling a large public energy company (dark green), where related critical devices are highlighted in green, business functions in blue, business processes in orange. This model was validated and verified to be reasonable by the company's IT experts. $n_N = 344$, $n_E = 754$. Visualized using Gephi [BHJ09].

(compare Section E.2.1, a mission dependency model is a Bayesian network) towards dynamic domains considering evolutions of states over time as commonly known as a dynamic Bayesian network (DBN). In DBNs values of random variables depend not only on current influences, but also on their respective history. An extension towards a dynamic probabilistic graphical model for dynamic mission impact assessment proposed in this section allows for time-dependent impacts, e.g., decaying impacts, evolving mission impact analyzes, and retrospective considerations of potential sources of impacts *inside* a network allowing for forensic analyzes.

So far in this article, we presented how mission dependency models, resource dependency models and impact models are obtainable, validatable and combinable towards one (static) probabilistic graphical model. Let R be a resource dependency model, then it is a straight forward extension to introduce a dimension of time t into a time-dependent model representing a resource dependency model R^t for each timeslice t. In every R^t, each resource node $RN_i^t \in R^t$ is dependent on its predeccessor $RN_i^{t-1} \in R^{t-1}$ forming a dynamic probabilistic graphical model (DPGM) as shown in Figure E.14. Respectively, at every timestep t a mission dependency model M^t is formed, whose business critical functions are dependent on some business critical resources in R^t. Likewise, at every timeslice some business resources are

threatened directly by some observed or unobserved shock events in SE^t. However, as described in Section E.3 a well-defined semantics is required for such a DPGM, which, due to the (potentially) cyclic nature of a resource dependency model, are not immediately given by classic DBN semantics.

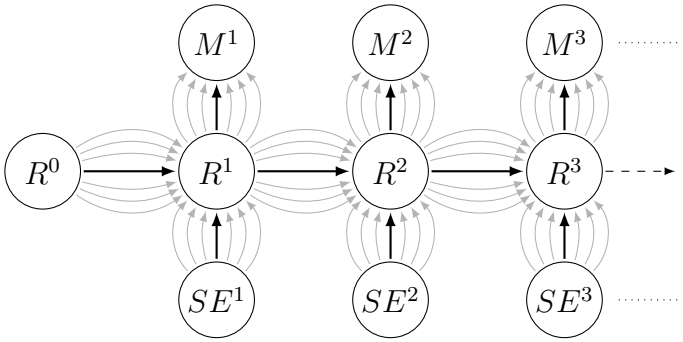


Figure E.14: An extension of the presented probabilistic graphical model for mission impact assessment towards a dynamic assessment, considering history states of (potentially impacted) nodes. This is beneficial for applications, where interactions of nodes are rapidly changing over *time* and a live tracking of impacts is required. Resource nodes RN_i^t of a ressource dependency model R^t are dependent on their predecessor $RN_i^{t-1} \in R^t$. R^0 represents an initial assumption about the potential impact state of nodes.

A resource dependency model is derived from (automatic) analyzes of communications between resource nodes. Considering such a dependency model over time allows for directly modeling each communication at time t as a dependency, i.e., an influence at time t. This means, with every communication there exists a probability of impact. For example, every time a significant amount of data is transferred from an, say, integrity impacted node to another, there exists a probability that the other node becomes impacted as well. The latter probability can, e.g., either be derived as described in Section E.2.2 or be derived by experts as described for the mission dependency model.

Still, a high rate of communications between of nodes as, e.g., present in IT infrastructures, is computationally too expensive. Considering an IT infrastructure as an information processing chain, it is reasonable to aggregate communications over a reasonable timeframe suited to an usecase. An aggregation significantly can reduce computational costs for obtaining inference results in DPGMs, but requires a careful consideration of indirect influences as [MM15b] has shown. Under the assumption of an information processing chain it is reasonable to assume that during one timeframe no cyclic communication occurs, i.e., a feedback loop in information processing is not finished in one timeframe. Then, in fact, the presented dynamic mission impact assessment model represents an ADBN by Motzek et al. in [MM15b] and [MM15a] and one obtains well-defined semantics. Based on an ADBN, dynamic mission impact assessments can be reduced to filtering and smoothing problems in ADBNs.

A reduction of mission impact assessment to problems in (A)DBNs has significant advances: One obtains the possibility to include *evidence* into a model, i.e., one is able to include information about actual observations of impacts on nodes. Then a *filtering* problem is formed by the problem of assessing the impact of any node at a time t given all obtained evidences so far. Observations, e.g., can origin from IDS alerts, antivirus scans, battle reports or plain-sight observations. A significant advantage of a (A)DBNs is that evidence is not only processed forwardly, but also backwardly. For example, if X is influenced by Y, and given an observed impact on node $X = +x$, an (A)DBN anticipates implications on Y by the observation of $X = +x$. Similarly, a *smoothing* problem is formed by an impact assessment of any node

at time k, given evidence obtained until time t. A solution to a smoothing problem delivers valuable information for forensic analyzes about intermediate states of mission impacts in retrospect, by including implications of future observations into an assessment.

Moreover, considering a Markov-1 property in ADBNs allows for *persisting* impacts. This is beneficial for situations, where a potential compromise on a node X already has lead to an impact on other nodes $\vec{Y}$, whose inferred impact will persists if even, e.g., a cleaning operation is later performed later on the origin node X. A Markov-1 ADBN is able to raise awareness for a potential impact on nodes $\vec{Y}$ and therefore on a higher goal, such as a mission, even though an original source has been eliminated.

E.6 Related Work

Mission modeling and mission impact assessment is an emerging field of research; and, naturally in new, viral research areas, employ ad-hoc solutions using algorithms involving fudge factors. While delivering early results and acclaimed solutions for mission impact assessment, a formal definition of an underlying problem is yet missing. Employed fudge factors in newly established algorithms lead to untraceable and spurious results demanding data driven validations. Unfortunately, large, standardized datasets for validation are yet missing for mission impact assessment and in the following presented work. Ad-hoc solutions can deliver acceptable results, if an expert is familiar with an employed solution and understands the nature of setup requirements, but frequently involves unvalidatable and unverifiable "black magic" (non-context-free). By resorting to a well-defined and understood mathematical problem, we use already validated and established approaches. In the following, we point out valuable approaches and ideas of related work, but withstep from a judgment.

Barreto et al. [dBBdCY13] introduce a well-understood modeling technique and use BPMN models to acquire knowledge. An impact assessment is based on various indexes and numerical scores, such as exploit index, impact factor, infrastructure capacity index, and graph distances. An assessment is solely based on direct impacts, leaving aside transitive impacts and/or defining a manual description of all dependencies between individual devices inside one organization, which is, in most of the cases an unfeasible process. Albanese et al. present in [AJJP13] a well-modeled formalism for complex inter-dependencies of missions as a set of tasks. Using numerical scores and tolerances in a holistic approach Albanese et al. focus on cost minimization. Buckshaw et al. [BPU$^+$05] propose a quantitative risk management by involving various experts and present a score-based assessment based on individual values and a standardization using a weighted sum.

Jacobson [Jak11] presents a well understood conceptual framework using interdependencies based on operational capacity. In this dependency model, impacts are propagated and reduce the operational capacity. [Jak11] uses self-defined metrics for propagating impacts through Boolean gates.

Further works focused solely on modeling. E.g., Goodall et al. [GDK09] focus on modeling and available data integration using ontologies but do not address an impact assessment. Another ontology-based approach is presented by D'Amico et al. in [DBGW10] and identifies multiple experts while noting that, e.g., system administrators are not capable of understanding an organization's missions.

In terms of (probabilistic) approaches towards assessments of impacts caused by vulnerabilities and attacks, probabilistic models have been researched by Wang et al. [WIL$^+$08], Liu et al. [LM05] or Xie et al. [XLO$^+$10]. However, Wang et al. base their work on attack graphs and do not consider imperfect knowledge. Xiep et al. and Liu et al. are significantly limited by the lack of supporting cyclic dependencies and do not consider any mission impact relations. Other impact propagation approaches, e.g., by

Kheir et al. [KDCB+09] or Jahnke et al. [JTM07], claiming to handle such details, are not probabilistic based and degrade to a handcrafted propagation algorithm with arbitrary scores.

Notwithstanding, we were inspired by several aforementioned modeling ideas, such as using the BPMN standard and we considered different views from various experts. To the best of our knowledge, we contribute a novel, formalized, mathematical mission impact assessment to this emerging research area.

E.7 Conclusion

We presented a well-defined mathematical mission impact assessment, based on a probabilistic approach, without introducing score-based propagation algorithms returning spurious results.

We relied on the expertise of different experts and merged all views without losing information or forcing an expert into a knowledge field he cannot understand. All defined parameters are validatable and understandable locally, i.e., do not require a global view towards a complete system and algorithm. Based on an established mathematical model, we reduced mission impact assessment onto an well-understood problem in computer science. Experimental results demonstrate scalability of the approach such that large-scale network scenarios can be handed.

Further future work is dedicated to integrating the presented mission impact assessment into a fully automated cyber-defense system and to extend the work towards described time-dependent models. Examples 5 and 6 are closely linked to each other and motivate that an automated response system can be reduced to a mathematical minimization of the expected mission impact in our model.

Bibliography

[Ace16] Acea SpA. The Acea Group, 2016.

[AJJP13] Massimiliano Albanese, Sushil Jajodia, Ravi Jhawar, and Vincenzo Piuri. Reliable Mission Deployment in Vulnerable Distributed Systems. In *DSN Workshops 2013: 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop, Budapest, Hungary, June 24-27, 2013*, pages 1–8. IEEE, 2013.

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *ICWSM 2009: 3rd International Conference on Weblogs and Social Media, San Jose, California, USA, May 17-20, 2009*, 2009.

[BPU+05] Donald L Buckshaw, Gregory S Parnell, Willard L Unkenholz, Donald L Parks, James M Wallner, and O Sami Saydjari. Mission Oriented Risk and Design Analysis of Critical Information Systems. *Military Operations Research*, 10(2):19–38, 2005.

[dBBdCY13] Alexandre de Barros Barreto, Paulo Cesar G. da Costa, and Edgar Toshiro Yano. Using a Semantic Approach to Cyber Impact Assessment. In *8th Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax VA, USA, November 12-15, 2013*, pages 101–108, 2013.

[DBGW10] Anita D'Amico, Laurin Buchanan, John Goodall, and Paul Walczak. Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions, and Users. In *5th International Conference on Information Warfare and Security*, pages 8–9, 2010.

[GDK09] John R Goodall, Anita D'Amico, and Jason K Kopylec. Camus: Automatically Mapping Cyber Assets to Missions and Users. In *Military Communications Conference*, pages 1–7. IEEE, 2009.

[Hen88] Max Henrion. Practical Issues in Constructing a Bayes Belief Network. *International Journal of Approximate Reasoning*, 2(3):337, 1988.

[Jak11] Gabriel Jakobson. Mission Cyber Security Situation Assessment using Impact Dependency Graphs. In *FUSION 2011: 14th International Conference on Information Fusion, Chicago, Illinois, USA, July 5-8, 2011*, pages 1–8, 2011.

[JSW02] Somesh Jha, Oleg Sheyner, and Jeannette Wing. Two Formal Analyses of Attack Graphs. In *Computer Security Foundations Workshop*, pages 49–63. IEEE, 2002.

[JTM07] Marko Jahnke, Christian Thul, and Peter Martini. Graph based Metrics for Intrusion Response Measures in Computer Networks. In *LCN 2007: 32nd Annual IEEE Conference on Local Computer Networks, Clontarf Castle, Dublin, Ireland, October 15-18, 2007*, pages 1035–1042, 2007.

[KDCB+09] Nizar Kheir, Hervé Debar, Nora Cuppens-Boulahia, Frédéric Cuppens, and Jouni Viinikka. Cost Evaluation for Intrusion Response Using Dependency Graphs. In *Network and Service Security, 2009. N2S'09. International Conference on*, pages 1–6. IEEE, 2009.

[Lan13] Ralph Langner. To Kill a Centrifuge. A Technical Analysis of What Stuxnet's Creators Tried to Achieve. Technical report, 2013.

[LM05] Yu Liu and Hong Man. Network Vulnerability Assessment Using Bayesian Networks. In *Defense and Security*, pages 61–71. International Society for Optics and Photonics, 2005.

[MM15a] Alexander Motzek and Ralf Möller. Exploiting Innocuousness in Bayesian Networks. In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, ACT, Australia, November 30 - December 4, 2015*, pages 411–423, 2015.

[MM15b] Alexander Motzek and Ralf Möller. Indirect Causes in Dynamic Bayesian Networks Revisited. In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, pages 703–709, 2015.

[MMLD15] Alexander Motzek, Ralf Möller, Mona Lange, and Samuel Dubus. Probabilistic Mission Impact Assessment based on Widespread Local Events. In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, pages 16–22, 2015.

[MTT+10] Scott Musman, Aaron Temin, Mike Tanner, Dick Fox, and Brian Pridemore. Evaluating the Impact of Cyber Attacks on Missions. In *5th International Conference on Information Warfare and Security*, pages 446–456, 2010.

[OGA05] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A Logic-based Network Security Analyzer. In *14th USENIX Security Symposium, Baltimore, MD, USA, July 31 - August 5, 2005*, 2005.

[Owe13] Art B. Owen. *Monte Carlo Theory, Methods and Examples*. 2013.

[Pan13] Panoptesec DOW. Panoptesec Annex I, Description Of Work. In *Project Deliverables of the Panoptesec Collaborative Research Project on Dynamic Risk Approaches for Automated Cyber Defence, Grant Agreement No: 610416, ICT-2013.1.5, Trustworthy ICT, Version 4th September, 2015*, 2013.

[RKT07] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI 2007: 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2462–2467, 2007.

[SH13] Teodor Sommestad and Amund Hunstad. Intrusion detection and the role of the system administrator. *Information Management & Computer Security*, 21(1):30–40, 2013.

[WIL+08] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An Attack Graph-Based Probabilistic Security Metric. In *Data and Applications Security XXII, 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, London, UK, July 13-16, 2008*, pages 283–296, 2008.

[XLO+10] Peng Xie, Jason H. Li, Xinming Ou, Peng Liu, and Renato Levy. Using Bayesian Networks for Cyber Security Analysis. In *DSN 2010: IEEE/IFIP International Conference on Dependable Systems and Networks 2010, Chicago, IL, USA, June 28 - July, 2010*, pages 211–220, 2010.

F SELECTION OF MITIGATION ACTIONS BASED ON FINANCIAL AND OPERATIONAL IMPACT ASSESSMENTS

Notes *This article will be published at ARES 2016: 11th International Conference on Availability, Reliability and Security in Salzburg (Austria) in between August 31th and September 2nd 2016.*

Abstract

Finding adequate responses to ongoing attacks on ICT systems is a pertinacious problem and requires assessments from different perpendicular viewpoints. However, current research focuses on reducing the impact of an attack irregardless of side-effects caused by responses. In order to achieve a comprehensive yet accurate response to possible and ongoing attacks on a managed ICT system, we propose an approach that relies on a response system that continuously quantifies risks, and decides how to respond to cyber-threats that target a monitored ICT system. Our Dynamic Risk Management Response (DRMR) model is composed of two main modules: a Response Financial Impact Assessor (RFIA), which provides an assessment concerning the potential financial impact that responses may cause to an organization; and a Response Operational Impact Assessor (ROIA), which assesses potential impacts that efficient mitigation actions may cause on the organization in an operational perspective. As a result, the DRMR model proposes response plans to mitigate identified risks, enable choice of the most suitable response possibilities to reduce identified risks below an admissible level while minimizing potential negative side effects of deliberately taken actions.

F.1 Introduction

The impact of an event is defined as the magnitude of harm that is expected to be perceived by an organization as a result of the consequences from unauthorized disclosure, modification, destruction, unavailability, or loss of information [1]. It may be expressed relative to the nature of its consequences. A way of expressing such nature is to bind the consequences on security dimensions. Commonly admitted natures for impacts on Information Systems are: Confidentiality, Integrity and Availability (CIA).

Current research works focus on considering the impact of attacks [2–5], by evaluating their severity and consequences, leaving aside the impact of security actions in mitigating the effects of such attacks. However, the analysis of current cyber events should also consider the impact of potential mitigation actions as well as time, geographic space and affected elements [6].

We adopt a quantitative risk-aware approach that considers the likelihood of success of the detected attacks, their induced impact, and the cost and consequences of response plans onto a higher goal, e.g., a company or mission. The research presented in this paper represents a work in progress towards the development of a comprehensive and practical dynamic risk management response system.

Our model considers two approaches: A Financial Impact Assessment (FIA) and an Operational Impact Assessment (OIA). A response FIA (RFIA) relies on a Return On Response Investment (RORI) index and a

geometrical model (named attack volume) to estimate the impact of security incidents (e.g. intrusions, attacks, errors) in a financial perspective, and to deploy mitigation actions accordingly. A response OIA (ROIA) considers that mitigation actions, while highly effective, could lead to operational negative side-effects inside the network and therefore onto a mission. ROIA evaluates proposed response plans based on validatable local impact- and dependency-assessments of dependencies inside an organization's business- and IT-infrastructure. This is beneficial for applications, where highly critical missions and resources must be protected, without sacrificing missions in favor of security.

The rest of the paper is structured as follows: Sections F.2 and F.3 discuss preliminaries and theory of a financial- and operational-impact assessment. Section F.4 describes a proposed dynamic risk management response model based on financial- and operational impact-assessments. A real world application of the proposed system is presented in Section F.5 showing the applicability of the proposed model. Section F.6 discusses related work and we conclude in Section F.7.

F.2 Financial Impact Assessment

Cost sensitive metrics have been proposed as a viable approach to find an optimal balance between intrusion damages and response costs, and to guarantee the choice of the most appropriate response without sacrificing the system functionalities. Measurements are either absolute or relative. Absolute measurements use precise values that scale with a given unit (e.g. hundreds, thousands, millions, etc); whereas relative measurements are methods for deriving ratio scales from paired comparisons represented by absolute numbers [7]. Relative measurements are useful in obtaining an overall ratio scale ranking of the alternatives. If the ratio produces repeatable and consistent results, the model can be used to compare security solutions based on relative values [8]. Examples of these models include the Return On Investment (ROI) and all its variants [8–10].

For the scope of this article, we consider sets of individual actions performed as a response to an adversary. Sets of these actions are called response plans:

Definition 11 (Response Plan)**.** *A response plan RP is a vector of mitigation actions, representing individual actions to be performed as a response to an adversary or threat opposed to an organization.* ▲

The Return On Response Investment (RORI) is a quantitative model for cost sensitive response based on a financial comparison of the response plans [11, 12]. RORI is an adaptation of the Return On Security Investment (ROSI) index, that provides a qualitative comparison of response plans. The RORI index considers not only the intrusion impact but also the effect of response plans, as shown in Equation F.1.

$$RORI = \frac{(ALE \cdot RM) - ARC}{ARC + AIV} \cdot 100 \tag{F.1}$$

All parameters are defined as follows:

Definition 12 (Annual Loss Expectancy, ALE)**.** *ALE corresponds to the attack impact loss that an organization is exposed to in the absence of mitigation actions. ALE is expressed in monetary values (e.g., $/year) and depends directly on the attack's severity and likelihood. ALE includes the loss of assets (L_a), the loss of data (L_d), the loss of reputation (L_r), the legal procedures (LP), the loss of revenues from clients or customers (L_{rc}), as well as other losses (L_o), contracted insurances (Ins), to be multiplied by the annual rate of occurrence of the attack (ARO), i.e.,*

$$ALE = (L_a + L_d + L_r + LP + L_{rc} + L_o - Ins) \cdot ARO\,. \qquad ▲$$

Definition 13 (Annual Infrastructure Value, AIV)**.** *AIV represents the fixed costs that are expected to be perceived by an organization regardless of the implemented mitigation action. AIV is strictly positive and is expressed in monetary values (e.g., $/year). It includes the following costs: equipment costs (C_e), personnel costs (C_p), service costs (C_s) and other costs (C_o), as well as the resell value (V_r), i.e.,*

$$AIV = C_e + C_p + C_s + C_o + V_r \ . \qquad \blacktriangle$$

Definition 14 (Risk Mitigation, RM)**.** *RM refers to the risk mitigation associated with a given mitigation action. RM takes values between zero and one hundred percent (i.e. 0% $\leq$ RM $\leq$ 100%). In the absence of mitigation actions, RM equals 0%. RM is computed as the product of the Mitigation Coverage (MC, which is the percentage of the attack covered by the mitigation action) by the Effectiveness Factor (EF, which is the percentage of reduction of the total incident cost given the enforcement of the mitigation action), i.e.,*

$$RM = MC \cdot EF \ . \qquad \blacktriangle$$

Definition 15 (Annual Response Cost, ARC)**.** *ARC refers to the costs associated to a given mitigation action. ARC is always positive and expressed in monetary values (e.g., $/year). It includes direct costs such as the cost of implementation (C_{impl}), the cost of maintenance (C_{maint}), as well as other direct costs (C_{od}) and indirect costs (C_i) that may originate from the adoption of a particular mitigation action, i.e.,*

$$ARC = C_{impl} + C_{maint} + C_{od} + C_i \ . \qquad \blacktriangle$$

Considering a RORI index alone, the best candidate response set is represented by a maximal positive RORI index.

F.3 Operational Impact Assessment

An operational impact assessment is used to address potential impacts onto a higher goal, from widespread events which impact local operational capabilities. For example, a local impact caused by an event on a distant node, might lead to a causal chain of operational failures, leading to an impact on a company. Understanding these impacts is a pertinacious problem and current work uses adhoc solutions based on handcrafted algorithms. While such approaches deliver early results, their assessments need to be verified and validated by large amounts of data—which is not always available.

Motzek et al. introduce an approach towards OIA based on a probabilistic graphical model in [26], which defines a well-understood problem on which an OIA can be reduced. By resorting to a probabilistic model, the use of conditional probability distributions allows for local views on assessments, without a need to understand a specific use case nor any algorithmic properties. It is this local view, which allows for a validation of defined data. This means, assessments from experts can be used directly without global normalization factors and experts are not forced into expertise which they can not understand.

The following sections introduce a view on OIA from three different perspectives, each defining one dependency model as a probabilistic graphical model of random variables and respective dependencies.

Remark 5 (Impact)**.** *An abstract term of "impact" is used in this work in the sense of "not operating as fully intended". The underlying meaning of "intended operation" lies in an use case of the model.* ▲

F.3.1 Mission Dependency Model (Business View)

Motzek et al. [26] extend a model by Jakobson [22] and model mission dependencies as shown in Figure F.1 as a graph of *mission nodes* (MN). A *company* is dependent on its *business processes*. A business process is dependent on one or more *business functions*, which are provided by *Business resources*. Figure F.1 shows a dependency graph of business relevant objects for a small company consisting of two business processes, requiring a total of four functions provided by four resources.

Dependencies are represented by local conditional probability distributions (CPDs) modeling probabilities of failure, given dependances fail. For example, the probability of business-function BF_1 (see Figure F.1), say, "provide access to customer data", failing, given required business-resource A, e.g., "customer-data-frontend", fails is 90%. [26] argues that the meaning of local conditional probabilities are understandable using common-sense (e.g., "in 9 out of 10 cases, customer data were not accessible for employees during frontend-server maintenance") and that the (numerical) assessment can be directly validated by either an expert or through ground-truth.

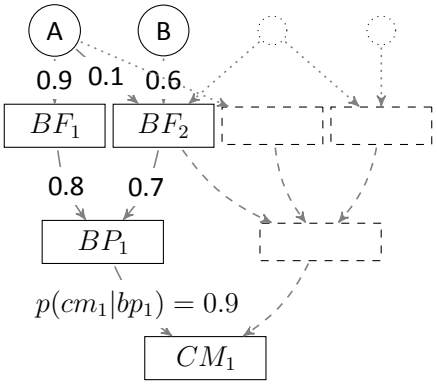


Figure F.1: Mission Dependency Model. Values along edges denote individual conditional probability fragments.

Definition 16 (Probabilistic Preliminaries)**.** *A node of a probabilistic dependency model is a random variable, denoted as capital X. Every random variable is assignable to one of its possible values $x \in$* dom(X)*. Let $P(X = x)$ denote the probability of random variable X having x as a value. For the case* dom$(X) = \{true, false\}$ *we write $+x$ for the event $X = true$ and $\neg x$ for $X = false$.* ▲

The event $+x$ represents the case that node X is operationally *impacted* and $\neg x$ that is operating as fully intended, i.e., no impact is present.

Definition 17 (From dependencies to distributions)**.** *Single dependencies of a random variable Y on X are modeled as* individual *conditional probability $p(x|y)$ and $p(x|\neg y)$. Such individual conditional probabilities are* fragments *of a complete CPD and are therefore denoted in lowercase. To acquire the local CPD $P(X|\vec{Y})$ of node X from* all *its fragments $p(X|Y)$ of all dependent nodes $Y \in \vec{Y}$, [26] employs a non-leaky noisy-or combination function as described in [23, 24].* ▲

With Definition 17, a mission dependency model is a Bayesian network, whose semantics is defined by the joint probability distribution over all mission nodes, i.e., random variables, as the product of all local defined CPDs.

Business resources are part of an infrastructure perspective and—from an operational view—might be irrelevant, but are identified to be business critical by a business expert. Notwithstanding, such an assessment might be inaccurate, which is why transitive impacts must be considered. For example, a web-service might be identified as a business critical resource; it can not be expected that an underlying distributed computing cluster is identified to actually provide this web-service. The following resource dependency model covers these dependencies.

F.3.2 Resource Dependency Model (Operation View)

Critical resources identified in a mission dependency model are *dependent* on further resources. Likewise, if a dependent resource is threatened, the identified critical resource might be threatened *transitively* as well. An operation expert, unlike a business expert, has an expertise to understand such dependencies, which we cover in an resource dependency model. The resource dependency model models dependencies between individual resources, which can be, e.g., individual ICT servers, ICS devices, software components or, in other use cases, manufacturing robots, suppliers, soldiers or vehicles. A "Bayesian" approach is followed as before, meaning that every dependency between two resources represents a local conditional probability of impact, if the dependence is impacted, as shown in Figure F.2.

[26] argues that assessing resource dependencies is not manageable by hand. Complex operation structures render a manual dependency analysis infeasible and error prone. Further, dynamically adjusting infrastructures (e.g., as found in IT cloud use cases) make it even unknown to an expert to identify exact dependencies. However, [26] argues that an expert is able to validate a presented infrastructure dependency model for plausibility. Therefore, a solution based on heuristics from exchanged information amounts are proposed to obtain a resource dependency model, for which we present an example in Section F.4.2.

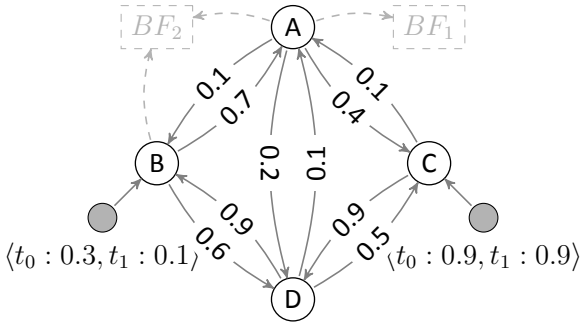


Figure F.2: A minimalistic resource dependency model. Conditional probability fragments are marked along the edges. Grey nodes represent external shock events leading to local impacts on resources. Connections to the mission dependency model are sketched in dashed gray.

F.3.3 Local Impacts (Security View)

Nodes of a resource dependency model might threatened directly by, so-called, external shock events. A security expert has the expertise to assess the local consequences on a node, given the presence of an shock event, e.g., the presence of a vulnerability or a direct shutdown of a node. An external shock event $SE \in \vec{SE}$ is a random variable and is present ($+se$) or not present ($\neg se$), for which a prior random distribution $P(SE)$ is defined. Every node X might be affected by one or more external shock

events $\vec{SE}$. Accordingly, the presence of an external shock event can be known or can be unclear and is assessed probabilistically through its prior random distribution $P(SE)$. The set of observed external shock events (known presence) is a set of instantiations $\vec{se}$ of observed random variables $\vec{SE}_O \subseteq \vec{SE}$. In the case that an external shock event is present (se), there exists a probability of it affecting a node X, expressed as a conditional probability fragment $p(+x|+se)$. If an external shock event exists and it is not inhibited, [26] speaks of a *local impact* on X. In the case that the external shock event is not present, i.e., $\neg se$, it does not affect random variable X. Every individual conditional probability fragment contributes to a random variable's CPD in the same way as a dependance on other nodes.

Definition 18 (Temporal Aspects)**.** *[26] defines a temporal aspect of an external shock event. In an abstract timeslices an effect of an external shock event changes. Every abstract timeslice represents a duplicate of the network- and mission dependencies with a different set of local conditional probabilities and prior probabilities of shock events. A time-varying probability is denoted as a sequence $\langle t_0 : p_0, \ldots, t_T : p_T \rangle$, with $T + 1$ abstract timeslices. In every abstract timeslice i, varying probabilities take their respective conditional or prior probability p_i defined for its timeslice t_i.* ▲

Note that a security expert does neither need to have any expertise in dependency analyses nor in business process analyses. An assessment of potential impacts is performed using a local, causal, view on resources and direct causes as external shock events.

F.3.4 Mathematical Mission Impact Assessment

To summarize, one probabilistic graphical model is defined by a mission dependency network, a resource dependency network and a set of external shock events with associated local impacts threatening nodes (or random variables) defined by the resource dependency network. As resource nodes are dependent on each other, a threatened node might again threaten another node, which leads to a global "spreading" of impacts induced by external shock events. In the end, there exists a probability that even a business process or the complete modeled company (mission) is threatened transitively by various external shock events, which is what we call the mission impact assessment.

Definition 19 (Mission Impact Assessment, MIA)**.** *The probability of a mission node MN being impacted, is defined as the conditional probability of MN being impacted $+mn$ given all observations of external shock events $se \in \vec{se}$, i.e. $P(+mn|\vec{se})$, where the effects of local impacts due to $\vec{se}$ are mapped globally based on mission-dependency and resource-dependency graphs. The mission impact assessment is therefore defined as the problem of obtaining $P(+mn|\vec{se})$, for all mission nodes MN defined defined in the mission dependency model.* ▲

Probabilistic inference is generally known to be NP-hard, and an exact solution for the MIA problem is only obtainable in small toy domains. However, approximate inference techniques are a valuable alternative for probabilistic inference. To obtain an algorithm determining an approximate solutions to the MIA problem, one can see the probabilistic model as a probabilistic logic program, where every "path" $w_i^{MN} \in \vec{w}^{MN}$ from an external shock event $SE \in \vec{SE}$ to the mission node MN is a conjunction of Boolean random variables and is a sufficient proof for satisfying $\{MN = true\} = +mn$. Due to the noisy or assumptions, $\vec{w}^{MN}$ then represents a disjunction of conjunctions. Every proof w_i^{MN} exists with a probability $P(w_i^{MN})$, where $P(w_i^{MN})$ is the product of all probabilities in this proof. Let $\mathbf{P}(w_i^{MN})$

denote the probability viewed as a set. $P(+mn|\vec{se})$ is then the probability that at least one proof holds, or rather, the probability that the disjunction of conjunctions is satisfied, i.e.

$$P(+mn|\vec{se}) = \bigcup_i \mathbf{P}(w_i^{MN}) = P(\vec{w}^{MN}) = P(\{\bigvee_i w_i^{MN}\}) \,,$$

where not all $\mathbf{P}(w_i^{MN})$ are disjoint. Calculating $\bigcup_i \mathbf{P}(w_i^{MN})$ is also known as the probabilistic satisfaction problem and is also used in the Problog reasoning framework [28]. To reduce computational complexity, a search for all "paths" $w_i^{MN} \in \vec{w}^{MN}$ can be limited to a fixed depth, e.g., using a depth-limited depth-first search. It is reasonable to limit a depth to an average path length in a graph to at least visit every node once, i.e., to at least include every external shock event once.

A probabilistic MIA $P(+mn|\vec{se})$ directly originates from all defined dependency-models and represents an inference problem in a probabilistic graphical model. Therefore, [26] argues that if locally defined dependency-models are validated to be correct, an obtained impact assessment $P(+mn|\vec{se})$ is validated, too.

F.4 Dynamic Risk Management Response System

We developed a system that proposes response possibilities to mitigate identified risks, enable choice of the most suitable response possibilities to reduce the identified risks below an admissible level, and then, compute the mitigation actions to be deployed on monitored or protected ICT systems. We adopt a quantitative risk-aware approach that provides a comprehensive view of the threats, by considering, (i) the likelihood of success of the considered attacks, (ii) their induced impact, and (iii) the cost and impact of the possible responses. Our dynamic response model is composed of two main modules: the Response Financial Impact Assessor (RFIA), and the Response Operational Impact Assessor (ROIA), as shown in Figure F.3.

The RFIA receives input data regarding the severity and likelihood of the potential threats, benefits and cost of mitigation actions, default security policies, and all elements that could be affected in the exploitation of the threat (e.g., users, channels, resources). The RFIA performs the evaluation of individual and combined mitigation actions and creates response plans based on the selected candidates. Such plans are sent to the ROIA for evaluation. For every proposed response plan, the ROIA provides an operational impact assessment.

Assessed response plans are propagated to an visualization module and to an policy enforcement point, which transforms mitigation actions into security policies. A response plan selection strategy proposed in this paper used to assist a security operator in selecting the most suitable response plan.

F.4.1 Response Financial Impact Assessor (RFIA)

The Response Financial Impact Assessor quantifies the level of benefit perceived per response plan on a financial basis. It provides an assessment concerning the potential financial impact that a given response plan may cause to an organization. Response plans represent proposed mitigation of the assessed risks and are assumed to be composed of one or more mitigation actions. The RFIA is composed of two main components: the Return On Response Investment (RORI), and the Attack Volume (AV), as depicted in Figure F.4.

From Figure F.4, the RORI interface requests the RORI Engine to perform the evaluation of mitigation actions (e.g., individual and combined) for a given threat scenario. The RORI Engine requests the input

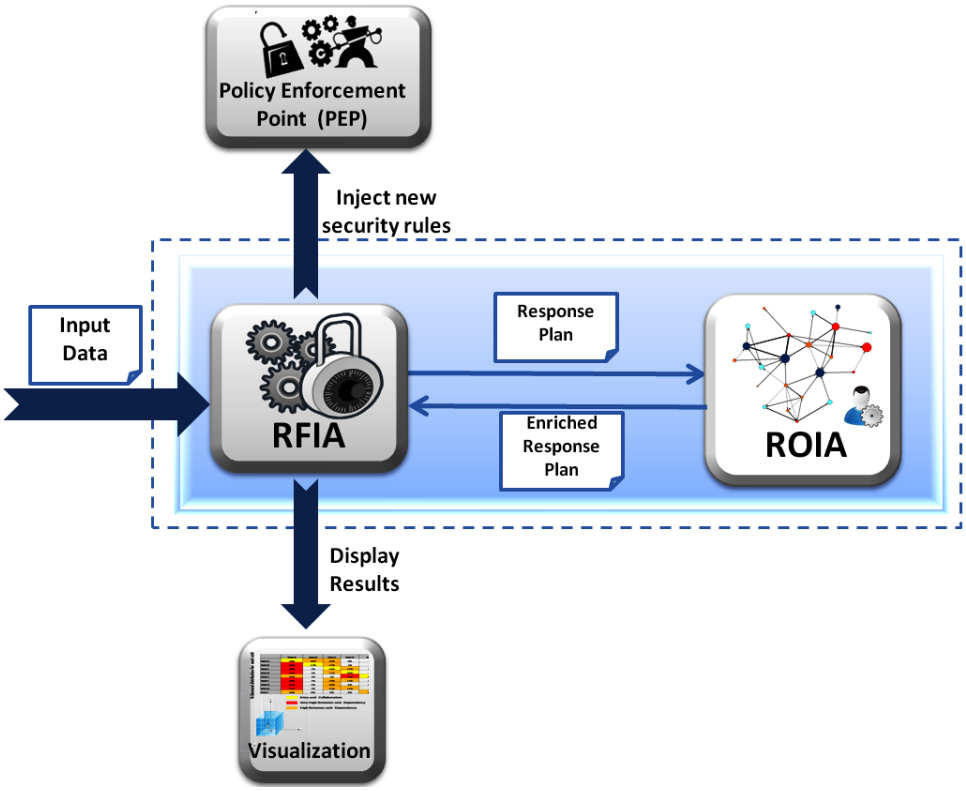


Figure F.3: Internal block diagram for the DRMRS components

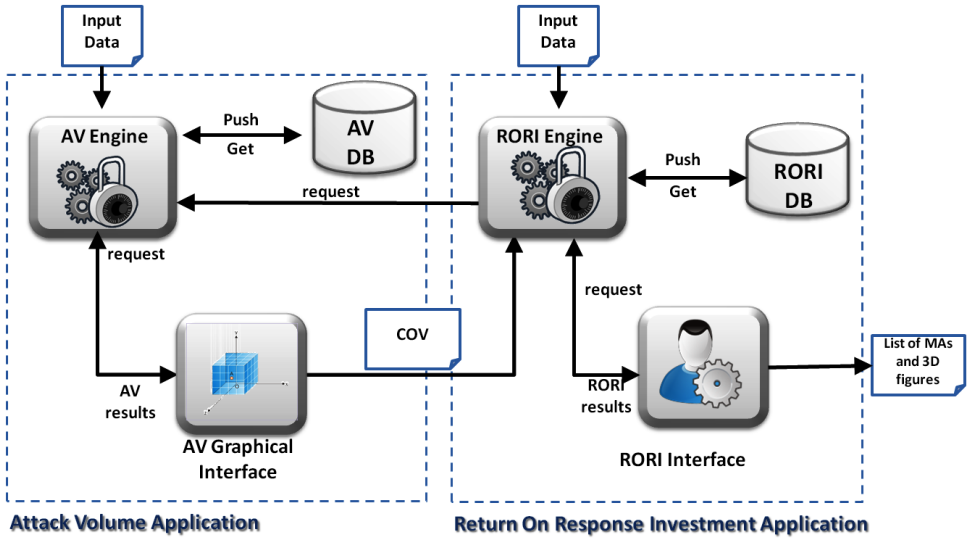


Figure F.4: Internal block diagram for the RFIA component

parameters (i.e., ALE, AIV, ARC, RM) to the RORI database through a get command. If the ALE or the RM are missing for that particular security incident, the RORI engine will request the AV Engine to perform the attack volume evaluation and to provide the corresponding values.

The AV Engine requests the input data to particular services (e.g., LDAP, databases, ACL, servers) and retrieves the associated RCU (Resource-Channel-User) information in order to calculate its volume and plot its graphical representation. The retrieved RCU data is stored in the AV database and displayed in

the AV interface.

Upon reception of all the parameters, the RORI engine stores them in the RORI database through a push command, and performs the evaluation of the authorized mitigation actions.

F.4.1.1 Return On Response Investment (RORI)

is a relative index that indicates the level of benefit perceived if a given mitigation action is implemented. The input parameters for the RORI calculation are of two kinds: fixed parameters include the Annual Infrastructure Value (AIV), which depends on the system, and the Annual Loss Expectancy (ALE), which characterizes the intrusion or attack; variable parameters include the Risk Mitigation (RM) and Annual Response Cost (ARC) which express the costs related to a mitigation action. RORI is calculated according to Equation F.1.

The RORI index is used to evaluate optimal plans, by ranking them as a trade-off between their efficiency in stopping potential attacks, and their ability to preserve, at the same time, the best service to legitimate users. Details on the estimation of each parameter composing the RORI model can be found in [12].

F.4.1.2 Attack Volume (AV)

is a graphical tool that evaluates the impact of one or multiple attacks and/or mitigation actions over a specific target. The representation of each attack is performed in a three-dimensional coordinate system i.e., user account (Acc), channel (Ip-Port), and resource (Res). The same coordinates include also system assets and potential mitigation actions. The projection of the three axis in our coordinate system generates a parallelepiped in three dimensions. The resulting volume is computed as the product of the axes contribution to the execution of the incident, i.e.,

$$AV(A) = Co_{Acc}(A) \times Co_{Ip-Port}(A) \times Co_{Res}(A) \tag{F.2}$$

The axis contribution is determined as the sum of the product of each set of axis category (e.g., user account type, port class, resource type, etc.) by its associated weighting factor. Each category within the axis contributes differently to the volume calculation. The weighting factor corresponds to the severity of a given category based on the CARVER[1] methodology [13].

The volume calculation requires the computation of the contribution of each axis represented in the coordinate system. This contribution is determined as the sum of each set of axis entities (e.g., user account type, port class, resource type) times its associated weighting factor (that results from the implementation of the CARVER methodology), i.e.,

$$Co_{Axis}(S) = \sum_{i=0}^{n} Count(E \in Type_{Axis}(S)) \times WF(Type_{Axis}(S)) \,. \tag{F.3}$$

The attack volume interface provides a 3D view of the complete attack scenario, making it possible to calculate the impact of multiple attacks that originate simultaneously in the system. In addition, we are

[1] A multi-criteria methodology that considers Criticality, Accessibility, Recuperability, Vulnerability, Effect, and Recognizability in the evaluation

able to compute the coverage of such attacks in the system, and the level of coverage for one or more response plans against the detected attack(s). Details on the computation of the system, attack and countermeasure volumes can be found in [14].

F.4.2 Response Operational Impact Assessor (ROIA)

The ROIA is divided into three components, which acquire, define and evaluate all information needed for an response operational impact assessment based on the described probabilistic model.

F.4.2.1 Network Dependency Analyzer

In order to obtain a resource dependency model automatically, we propose a module which consecutively captures traffic inside an organization and analyses statistics in them based on a heuristic. In our use case, a resource dependency model consists of a medium sized ICT environment, in which some ICT devices also represent gateways to an industry SCADA system. Further, it can be assumed that every device drives one purpose. This allows for a simple heuristic on exchanged information amounts to obtain a plausible resource dependency network, explained at the following simple example: A workstation X consuming different query results from multiple databases distribute gained and processed information from such queries to other devices. The percentage of received traffic $T_{Y_i,X}$ from every database Y_i towards the total received traffic gives a good guideline for the conditional dependency between them as $p(x|y_i) = \frac{T_{Y_i,X}}{\sum_i T_{Y_i,X}}$. Depending on a network or company characteristics other heuristics might be appropriate, e.g. derivation from a mean received amount of data or a mapping onto a σ distribution. As long as no irrelevant information is consumed and distributed to other resources, this heuristic results in a plausible resource dependency model. As part of the ROIA, we implement this approach in an automatic module, periodically capturing traffic and analyzing obtained results, which are presented in Section F.5.

F.4.2.2 Local Impact Definition

The introduced probabilistic mission impact model is based on general external shock events. In order to obtain a *response* OIA, a response plan must be transformed to external shock events. Every mitigation action inside a response plan represents a potential cause for local harm. For example, a shutdown of a node X might cause other transitively dependent nodes to not work as intended, i.e., become impacted. Assessing the global effects of a local action is intuitively not possible and is the goal of an ROIA. However, local assessments are validatable and can even be grounded on common sense: Given one shuts down a node X, the probability that it will be impacted, i.e., not work as intended, is 100%: $p(+x|+shutdown_x) = 1$. We extended [26]'s proposed external shock event transformation from response plans:

Definition 20 (Response Plan Side Effects)**.** *We define external shock events by using three abstract temporal timeslices: t_0 representing a short-term impact, t_1 representing a mid-term impact, and t_2 representing a long-term impact. If a node is* shut down *(+se: the external shock event is present) it is easy to assess a probability of local impact to be 1. This means, $p(+x|+se) = \langle t_0 : 1, t_1 : 1, t_2 : 1\rangle$. Likewise,* restart*ing a resource has the same effect as a shutdown in t_0, and might likely lead to hardware failure during reboot in a mid-term t_1, but will locally not cause conflicts in a long-term: $p(+x|+se) = \langle t_0 : 1, t_1 : 0.6, t_2 : 0\rangle$.*

Employing a patch *on a node X might produce collateral damage as well. During installation of the patch, there exists a (low) probability of immediate conflict, e.g., a flat assumption of 10% or a measure published by a software vendor. In a mean time, a patch might enforce a reboot of a resource. This leads to a temporal shutdown and might lead to hardware failure. Finally, after a successful reboot, a replacement of hardware, and/or a restore of a previous backup, the network device will fully resume its operational capability. Therefore, $p(+x|+se) = \langle t_0 : 0.1, t_1 : 1.0, t_2 : 0.0 \rangle$. We argue that every installation, update or change of software can be modeled from an impact perspective as a patching operation.*

Like software is exchanged by a patch, hardware can be reconfigured as well. A reconfiguration *is likely to enforce a reboot, if an exchanged component is not hot swappable. Therefore, we assume the same local impact as induced by a reboot.* ▲

Further examples for shock events are given by Motzek et al. in [26].

F.4.2.3 Monte-Carlo Evaluator

As mentioned before, an exact calculation of $\bigcup_i \mathbf{P}(w_i^{MN})$ is possible by the inclusion and exclusion principle and the Sylvester-Poincaré equality, but is exponentially hard due to the subtraction of all overlapping sets and is therefore not practical. We therefore approximate a solution to the MIA problem by the use of an approximate inference technique.

For every mission node MN, there exists a Boolean formula $\vec{w}^{MN}$ as a disjunction of conjunction over Boolean random variables $\vec{B}$. However, Boolean random variables in $\vec{B}$ take their respective truth value according to a probability distribution. To approximate $\bigcup_i \mathbf{P}(w_i^{MN})$, i.e., to find an approximate solution to the MIA problem, a complete instantiation of all Boolean variables $\vec{B}$ is drawn by sampling every Boolean variable according to its distribution, and $\vec{w}^{MN}$ is checked for satisfaction. Repeating this process n times, where n^+ times a satisfaction was found, approximates $P(+mn|\vec{se})$ by n^+/n. Our results show that an upper three-sigma bound of expected error $\bar{E}$ is obtained by $\bar{E} = 0.775 \cdot \sqrt{n}^{-1}$. A detailed description and evaluation is given in [26] and left out for brevity in this paper.

F.4.3 Selection of Response Plans

Both, RFIA and ROIA, perform impact assessments of a proposed response plan as a collection of individual mitigation actions. Due to their nature, a financial impact assessments (using a RORI index) and an operational impact assessments are performed from perpendicular perspectives: On the one hand, the less invasive a response plan is, the less it can potential cause collateral damage. On the other hand, a minimally invasive response plan, will not significantly reduce a risk. It is the novel advantage of the proposed DRMRS of being able to combine both assessments.

However, finding an optimal response plan in all dimensions defined by an OIA and a FIA is not trivial. The proposed FIA results in a linear, relative metric, i.e., assessments depend on a use-case and context and are only interpretable, evaluable and comparable during one evaluation of a set of response plans. Still, among one evaluation there exists a well-defined ordering. However, relative reference points are required for obtaining an absolute scale from one evaluation.

The proposed OIA is based on a probabilistic model resulting in a stable, absolute metric, e.g., an assessment of, say, 5% is understandable and interpretable independent of any context, use-case or

evaluation. For example, an OIA of 5% for a potential impact on a company, given a set of observed external shock event, is equivalent to a 5% of winning a lottery, given one plays the lottery, or a 5% probability of tossing a 1 on a twenty-sided cube. However, an OIA consists an n-dimensional vector representing a temporal diversity.

Due to a missing absolute scale in FIA and an assumed incomparability of temporal dimensions, an optimization goal by a defined cost function is not available. We therefore propose a selection of response plans based on a best compromise, i.e., a semi-optimal solution among all impact assessment dimensions, related to a Pareto optimum.

Definition 21 (Semi-optimal response plans)**.** *Let $\vec{RP}^d$ be a vector of proposed response plans, associated with a linearly scaled impact assessment of dimension d. Let $\dot{RP}^d \subseteq \vec{RP}^d$ denote the set of optimal proposed response plans in terms of dimension d. Let $\hat{RP}^d$ denote the assessment of the theoretical optimal response plan and let $\check{RP}^d$ denote the assessment of the theoretical worst response plan in terms of dimension d. Then, let $\dot{RP}^d_\varepsilon \subseteq \vec{RP}^d$ represent the set of semi-optimal response plans in terms of dimension d and easing factor $\varepsilon \in [0,1]$ representing the allowed deviation ε of the theoretical response plan range $|\hat{RP}^d - \check{RP}^d|$ from the evaluated optimal response plan $\dot{RP}^d$. Thus, $\dot{RP}^d_0 = \dot{RP}^d$ and $\dot{RP}^d_1 = \vec{RP}^d$.* ▲

Finding a best compromise among an n-dimensional impact assessment is therefore defined as finding the smallest semi-optimal set.

Definition 22 (Smallest semi-optimum)**.** *Let $\vec{d}$ be the vector of all impact dimensions. Then, the smallest semi-optimal set of response plans $\mathring{RP}$ is the set*

$$\mathring{RP} = \min_{\varepsilon} \left(\left\{ \bigcap_{d \in \vec{d}} \dot{RP}^d_\varepsilon \right\} \neq \emptyset \right) . \quad ▲ \tag{F.4}$$

As the ROIA represents an absolute metric, $\check{RP}^{ROI} = 1$ and $\hat{RP}^{ROI} = 0$. For the relative RFIA metric $\check{RP}^{RFI}$ and $\hat{RP}^{RFI}$ depend on $\vec{RP}^{RFI}$. If not all possibly allowed response plans are evaluated by the RFIA for performance criteria, $\check{RP}^{RFI}$ and $\hat{RP}^{RFI}$ are not uniquely identifiable and must be estimated by $\check{RP}^{RFI} = -1$ and $\hat{RP}^{RFI} = \dot{RP}^{RFI}$. This means, $\dot{RP}^{RFI}_\varepsilon$ might be too large. A selection of a response plan according to Definition 22 can efficiently be performed by using a binary search.

F.5 Use Case

This section studies an application of the proposed DRMRS in an infrastructure environment of an Energy Distribution Organization. The environment consists of a distributed network of remote terminal units (RTU) in energy stations of medium voltage (MV = 20,000 Volts) and high voltage (HV = 150,000 Volts). RTUs acquire data from electrical equipments (e.g., PLC, sensors, etc.), and send data to a supervisor terminal unit (STU) of the headquarter. The RTU network utilizes Supervisory Control and Data Acquisition (SCADA) protocols and is composed of over 13,000 energy stations, 6,000 of which are controlled by the STU.

In the absence of security compromise, operators review the security status of the Monitored System (SCADA and ICT environment). Security status indicators may note the presence of one or more system vulnerabilities due to known software security flaws as posted by publicly available vulnerability advisory

services. Attack paths from hypothetical attack sources to known mission critical systems are analyzed and the impact on critical business functions (e.g., energy distribution) is assessed resulting in a quantified risk assessment.

For testing purposes, we emulated the energy distribution organization (EDO) using the information shown in Table F.1.

Table F.1: Information of the EDO system

Dimension	Elements	Description	Q	WF
Resource	R1:R12	HV/MV Server	12	1-5
	R13:R16	HV/MV Front End	4	4
	R17:R22	HV/MV Gateway	4	4
	R23:R56	Routers	34	3-4
	R57:R63	Human-Machine Interface	6	2-3
	R64:R363	Remote Terminal Unit	300	5
	R364:R365	Firewall	2	2
	R366	PC	1	2
	R367:R368	IDS	2	2
Channel	Ch1:Ch2	Public IP address	2	3
	Ch3:Ch302	Private IP address	300	2
	Ch303:Ch698	UDP Port	396	1-5
	Ch699:Ch1712	TCP Port	1014	3-5
User Account	U1:U30	Basic Operator	30	1
	U31:U38	Advanced Operator	8	4
	U39:U52	High Voltage Operator	14	3
	U53:U70	Medium Voltage Operator	18	2
	U71	Supervisor	1	5

From Table F.1, we organize the information of the EDO according to their nature (dimension). We have for instance, servers, firewalls, IDs, etc as resources; IP addresses and port numbers as channels, and operators as user accounts. Depending on the type of element and their importance to the mission of the organization, we assign a weighting factor. A basic operator is assigned a WF=1, whereas an advanced operator has a WF=4, and a supervisor has a WF=5. For those cases where the category regroups elements of different types (e.g., SCADA Servers, Web servers, NTP Server, etc are regroup as Servers), we assign a weighting factor for each type of element, going from one to five.

The annual infrastructure value for the EDO is equivalent to 11,379,800.00 €, which represents the cost of operation, license, maintenance and services incurred in a yearly basis for the regular operations of the organization. It considers the annual cost of all the policy enforcement points (PEPs) of the organization.

F.5.1 Threat Scenario

Security experts from the use case partner identified a threat, called “AS02”, which corresponds to a compromise of a specific target through vulnerability exploitation. More precisely, the threat will cause data corruption or leakage of a database in the ICT domain, e.g., from a file server.

There exists an attack vector via the ICT Network from, e.g., a vulnerable router “VR-08”, targeting first a Web server “Web-SRV”, second, a workstation “User-PC”, and third, a file server “File-SRV”.

F.5.2 Financial Impact Assessment

Threat AS02 has a severity defined as “serious”, which corresponds to 1,000,000 €, and a likelihood defined as “high”, which corresponds to a value of 12. The ALE is computed as 12,000,000 €/year. This threat has been associated to a set of mitigation actions. Combinations of associated mitigation actions form response plans that shall improve the security status of the monitored system (e.g., patch deployment, shutdown, restart, or other system reconfiguration). They are selected and executed by operators resulting in automated deployment of mitigation actions where possible (e.g., firewall reconfigurations) or otherwise issuing instructions to senior operators for follow-up deployment of actions (e.g., patch deployment). The detailed information of all authorized mitigation actions is shown in Table F.2.

Table F.2: Mitigation Action RFIA Evaluation

MA	Description	EF	COV	RM	ARC	Restriction	RORI
MA_1	Reconfig. V-R08	1.00	0.60	0.60	50	None	63.27
MA_2	Reconfig. Web-SRV	0.80	0.15	0.12	1,000	MA_{10}	12.64
MA_3	Reconfig. File-SRV	0.80	0.15	0.12	500	MA_{11}	12.65
MA_4	Patch Web-SRV	1.00	0.15	0.15	2,000	MA_{10}	15.8
MA_5	Patch File-SRV	1.00	0.15	0.15	500	MA_{11}	15.81
MA_6	Patch User-PC	1.00	0.10	0.10	500	MA_{12}	10.54
MA_7	Restart Web-SRV	0.01	0.15	0.00	50	MA_{10}	0.16
MA_8	Restart File-SRV	0.01	0.15	0.00	50	MA_{11}	0.16
MA_9	Restart User-PC	0.01	0.10	0.00	50	MA_{12}	0.11
MA_{10}	Shutdown Web-SRV	0.10	0.15	0.01	50	$MA_{2,4,7}$	1.58
MA_{11}	Shutdown File-SRV	0.10	0.15	0.01	50	$MA_{3,5,8}$	1.58
MA_{12}	Shutdown User-PC	0.10	0.10	0.01	50	$MA_{6,9}$	1.05

Table F.2 summarizes the information about mitigation actions that are authorized as a response to the specified threat AS02. ARC and EF of each security mitigation action were estimated based on expert knowledge and historical data. The RM value is calculated as the product of the EF and coverage (COV). A coverage is obtained using geometrical operations from the attack volume model. The RORI index is calculated using Equation F.1.

From the list of proposed mitigation actions, MA1 (Reconfiguration of V-R08) provides the highest RORI index. By taking this action, the risk is expected to be reduced to 60% (RM), resulting in a RORI index

of 63.27. Response plans for this threat are formed by combinations of all possible mitigation actions, considering those actions that are mutually exclusive (e.g.,MA_{10} can not be simultaneously implemented with MA_2, MA_4, and MA_7). All potential combinations, i.e., 797 response plans are evaluated and the best response plan results in a RORI index of $\hat{RP}^{RFI} = 97.1435$ with a combination of mitigation actions as $\langle MA_1, MA_2, MA_3, MA_4, MA_5, MA_6, MA_7, MA_8, MA_9\rangle$. The worst is represented by $\{\langle MA_7, MA_9\rangle \langle MA_8, MA_9\rangle\}$ with $\check{RP}^{RFI} = 0.21$.

The graphical representation of the best response plan vs. the evaluated threat is depicted in Figure F.5, where hashed lines represent threat AS02 and colored lines represent the mitigation actions.

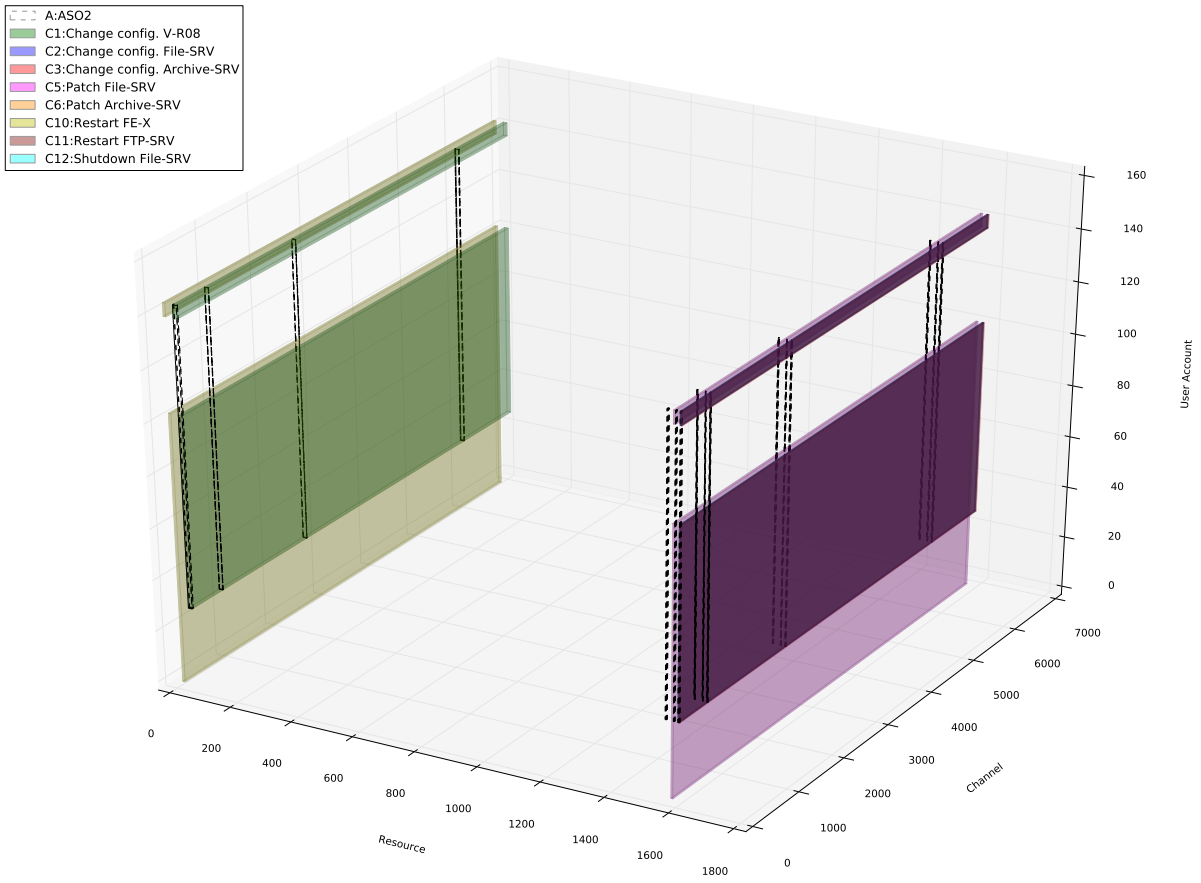


Figure F.5: Graphical representation of the threat and the best response plan.

F.5.3 Operational Impact Assessment

To perform a response operational impact assessment, a resource dependency model is needed. As described in Section F.3 a manual assessment is said to be infeasible and a solution based on a heuristic of exchanged traffic information was proposed in Section F.4.2. An evaluation of one week of traffic recordings inside the EDO resulted in a resource dependency model as shown in Figure F.6 consisting of 35 nodes and 66 edges (dependencies); 470 other nodes resembled insignificant dependencies and were removed for visualization and anonymization purposes. The average path length between two nodes resulted to be 2.5, s.t., a maximum search depth of 7 is likely to cover all paths.

Based on the resource dependency model and a mission dependency model defined by business experts from the use case partner, ROI assessments for all proposed response plans are evaluated. The mission dependency model for the EDO consists of four business processes and 26 identified critical resources, from which not all are yet simulated and not needed for the scope of this paper. For anonymization purposes and non-disclosure agreements, the mission dependency model can not be displayed here. A comparison between a RORI index and operational impact assessments in three temporal dimensions are given in Table F.3. Note how both lowest and highest probabilities of operational impact lead to extremely low RORI indices. In fact, Pearson's product-moment correlation coefficient between RORI and OI_0 and OI_1 for all evaluated response plan is ≈ 0.14 and between RORI and OI_2 even ≈ 0.01, showing that RORI and OI are almost uncorrelated.

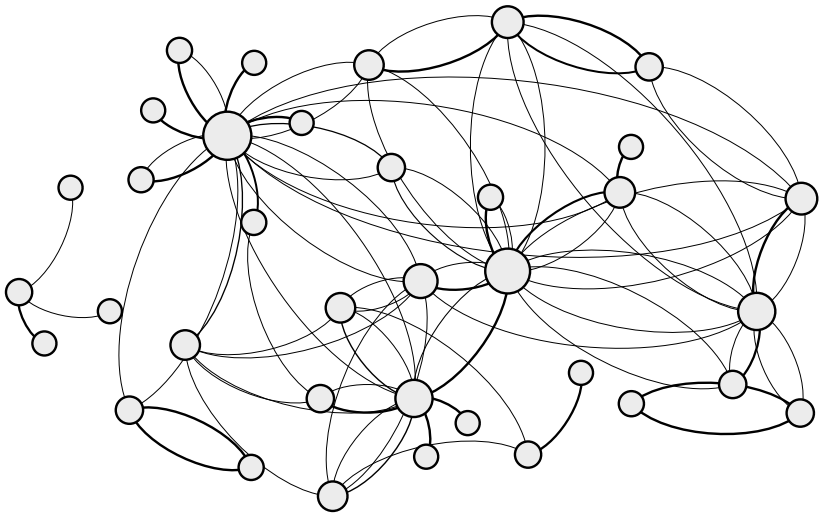


Figure F.6: Resource dependency model extracted from the use case partner. Thicker and darker edges represent higher dependency degrees. Visualized using Gephi [27].

Table F.3: Mitigation Action ROIA Evaluation. (*RORI given for reference*)

MA	Description	RORI	OI_0	OI_1	OI_2
MA_1	Reconfig. V-R08	63.27	4.2%	2.4%	0
MA_2	Reconfig. Web-SRV	12.64	6.6%	3.6%	0
MA_3	Reconfig. File-SRV	12.65	36.6%	22.2%	0
MA_4	Patch Web-SRV	15.8	0.6%	6.6%	0
MA_5	Patch File-SRV	15.81	3.6%	37.2%	0
MA_6	Patch User-PC	10.54	0.6%	6.6%	0
MA_7	Restart Web-SRV	0.16	6.6%	4.2%	0
MA_8	Restart File-SRV	0.16	36.6%	22.2%	0
MA_9	Restart User-PC	0.11	6.6%	4.2%	0
MA_{10}	Shutdown Web-SRV	1.58	7.2%	7.2%	7.2%
MA_{11}	Shutdown File-SRV	1.58	40.8%	40.8%	40.8%
MA_{12}	Shutdown User-PC	1.05	7.2%	7.2%	7.2%

F.5.4 Selection of Mitigation Actions

Judging from Table F.3 a good compromise is to singly deploying mitigation action MA_1, resulting in both a low probability of operational impact and being financially attractive in terms of RORI. The most financially attractive response plan $RP_R = \langle MA_1, MA_2, MA_3, MA_4, MA_5, MA_6, MA_7, MA_8, MA_9 \rangle$ with a RORI index of 97.1435, however, is assessed to bear almost the highest probability of operational impact with $\langle t_0 : 0.408\, t_1 : 0.402\, t_2 : 0.0 \rangle$. Note that an OI assessment of a response plan is *not* a linear combination of individual mitigation actions, as a "double count" of probabilities is not allowed and would lead to spurious results. In terms of lowest short-term (t_0) OI probability, MA_4 and MA_6 alone show to be dominant, and in mid-term (t_1) MA_1 alone is dominant. In a long-term perspective (t_2) a large set of response plans is dominant with a 0 probability of impact. Thus RP_R, MA_4 and MA_6 represent a Pareto optimal set. As proposed in Section F.4.3 we search for the best *compromise*: From Definition 22 one obtains the best semi-optimal response plan set $\mathring{RP} = \{\langle MA_1, MA_2, MA_4, MA_6, MA_7, MA_9 \rangle\}$ using $\varepsilon = 0.1475$, consisting of one response plan with an operational impact assessment of $\langle t_0 : 0.108\, t_1 : 0.09\, t_2 : 0.0 \rangle$ and a RORI index of 82.8514. This means, with a compromise of 14.75% of the theoretical optimum in every dimension from the evaluated optimum, a semi-optimal response plan is found.

Notwithstanding, one could normalize all impact dimensions to a range between $[0, 1]$, where 1 represents the best (i.e., 0 for operational impact and the best evaluated RORI index for financial impact) and 0 the worst assessment and then define an equally weighted cost function f used for selection of an optimal response plan. Following such approach, one selects: $RP_f = \langle MA_1, MA_2, MA_4, MA_6, MA_9 \rangle$ with $\langle t_0 : 0.102\, t_1 : 0.09\, t_2 : 0.0\, rori = 82.7732 \rangle$, which is different from $\mathring{RP}$. The following example clarifies the difference: Say, there exists another response plan RP^* with an assessment $\langle t_0 : 0.191\, t_1 : 0.0\, t_2 : 0.0\, rori = 82.7732 \rangle$, i.e., an assessment similar to the one of RP_f, but where an impact of dimension t_1 is moved to t_0 with a small difference. RP^* would be assigned an even better cost than RP_f by f, but the t_0 assessments differs by $\varepsilon = 18.5$% of the theoretical optimum from the evaluated optimum in dimension t_0 instead of 14.75% as $\mathring{RP}$ does.

F.6 Related Work

Current researches focus on considering the impact of attacks by evaluating their severity and consequences, leaving aside the impact of security actions in mitigating the effects of such attacks. Dini and Tiloca [2], for instance, propose a simulation framework that evaluates the impact of cyber-physical attacks, discusses the attack ranking process, and analyzes different mitigation actions. However, these latter are not considered in the calculation of the attacks' impact nor they are ranked according to their effectiveness in stopping or mitigating the attacks.

Kundur et al. [3], propose a paradigm for cyber attack impact analysis that employs a graph-theoretic structure and a dynamical systems framework to model the complex interactions amongst the various system components. The approach involves quantifying the effects of given classes of cyber attack, providing information on the degree of disruption that such class of attacks enable, and identifying sophisticated dependencies between the cyber and physical systems, but leaves aside the impact of mitigation actions in the attack's impact calculation.

Squoras et al. [5] present a qualitative assessment of the cyber attack impact on critical Smart Grid infrastructures. Authors evaluate the impact of DoS/DDoS attacks on data availability without considering mitigation actions in the assessment of the overall impact calculation.

In terms of operational impact assessment, probabilistic models have been researched as an adequate assessment of impacts or risks posed due to attacks or found vulnerabilities [15–17]. However, often imperfect knowledge is not considered [15] or dependency cycles pose a problem [17]. Other impact propagation approaches, able to handle such details, are not probabilistic based and degrade to a handcrafted propagation algorithm with arbitrary scores [18, 19].

Barreto et al. [20, 21] only consider direct impacts as approaches to mission modeling, leaving aside transitive impacts and/or defining a manual description of all dependencies between individual devices inside one organization, which is, in most of the cases an unfeasible process.

Our approach proposes the evaluation and selection of mitigation actions based on the financial- and operational-assessment of security events (e.g., attacks and mitigation actions). The response financial assessment considers the RORI as an index that ranks mitigation actions based on multiple factors (e.g., attack's impact, size of the infrastructure, cost and benefits of mitigation actions). The response operational assessment evaluates threats according to their nature expressed as local time-varying impacts and considers transitive impacts based on a well-defined probabilistic model, between an organization's missions and resources. The ultimate goal of our approach is to select the set of mitigation actions that provides the maximal positive financial gain and the minimal operational negative side-effect.

F.7 Conclusion

In this paper we have proposed an automatic response system, reacting to threats opposed on a company based on a multi-dimensional impact assessments. Two different impact assessment approaches have been incorporated, which seem to be conflicting at first sight: Every action taken in order to reduce a potential attack vector, bears a potential negative side effect that needs to be reduced.

Based on a multi-dimensional minimization proposal, we propose the choice of semi-optimal response plans that on the one hand bear the highest financial attractiveness on return on investment, and, on the other hand, bear the lowest probability of conflicting with a company's missions. This is beneficial for applications, where highly critical missions and resources must be protected, without sacrificing missions in favor of security.

Bibliography

[1] R. Kissel,*Glossary of key information security terms*, National Institute of Standards and Technology, U.S. Department of Commerce, 2011.

[2] G. Dini, M. Tiloca,*On simulative analysis of attack impact in Wireless Sensor Networks*, 18th Conference on Emerging Technologies & Factory Automation (ETFA), 2013.

[3] D. Kundur, X. Feng, S. Liu, T. Zourntos, K.L. Butler-Purry, *Towards a Framework for Cyber Attack Impact Analysis of the Electric Smart Grid*, International Conference on Smart Grid Communications (SmartGridComm), pp. 244–249, 2010.

[4] P. Su, X. Chen, H. Tang, *DoS Attack Impact Assessment based on 3GPP QoS Indexes*, 3rd International Conference on Innovative Computing Information and Control, 2008.

[5] K. I. Sgouras, A. D. Birda, D. P. Labridis, *Cyber Attack Impact on Critical Smart Grid Infrastructures*, Innovative Smart Grid Technologies Conference (ISGT), 2014.

[6] B. Roberts, *The Macroeconomic Impacts of the 9/11 Attack: Evidence from Real-Time Forecasting*, Working Paper, Homeland Security, Office of Immigration Statistics, 2009.

[7] T. L. Saaty, *What is relative measurement? The ratio scale phantom*, Mathematical and Computer Modelling Journal, vol. 17, number 4-5, pp. 1–12, 1993.

[8] W. Sonnenreich, J. Albanese, B. Stout, *Return On Security Investment (ROSI) – A Practical Quantitative Model*, Journal of Research and Practice in Information Technology, vol. 38, number 1, 2006.

[9] M. Jeffrey, *Return on Investment Analysis for e-Business Projects*, Internet Encyclopedia. Hossein Bidgoli Editor, vol. 3, pp. 211–236, 2004.

[10] Lockstep Consulting, *A Guide for Government Agencies Calculating Return on Security Investment*, Technical Paper, 2004

[11] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, H. Debar, *A Service Dependency Model for Cost-Sensitive Intrusion Response*, European Symposium on Research in Computer Security (ESORICS), pp. 626–642, 2010.

[12] G. Gonzalez-Granadillo, M. Belhaouane, H. Debar, G. Jacob, *RORI-based countermeasure selection using the OrBAC formalism*, International Journal of Information Security, Vol. 13(1), pp. 63–79, 2014.

[13] T. L. Norman, *Risk Analysis and Security Countermeasure Selection*, CRC Press, Taylor & Francis Group, 2010.

[14] G. Gonzalez Granadillo, J. Garcia-Alfaro, H. Debar, *Using a 3D Geometrical Model to Improve Accuracy in the Evaluation and Selection of Countermeasures Against Complex Cyber Attacks*, In Security and Privicay in Communication Networks, vol. 164, pp. 538–555, 2015.

[15] L. Wang, T.a Islam, T. Long, A. Singhal, S. Jajodia *An attack graph-based probabilistic security metric*, Data and Applications Security XXII. Springer Berlin Heidelberg, 283–296, 2008.

[16] L. Yu, H. Man *Network vulnerability assessment using Bayesian networks*. International Society for Optics and Photonics, 2005.

[17] P. Xie, J. Li, X. Ou, P. Liu, R. Levy, *Using Bayesian networks for cyber security analysis*, International Conference on Dependable Systems and Networks, pp. 211–220, 2010.

[18] N. Kheir, H. Debar, N. Cuppens-Boulahia, F. Cuppens, J. Viinikka *Cost evaluation for intrusion response using dependency graphs*, International Conference on Network and Service Security, 2009.

[19] J. Marko, C. Thul, P. Martini, *Graph based metrics for intrusion response measures in computer networks*, 32nd IEEE Conference on Local Computer Networks, 2007.

[20] A. Barreto, P. Costa, E. Yano, *A Semantic Approach to Evaluate the Impact of Cyber Actions to the Physical Domain*, 7th International Conference on Semantic Technologies for Intelligence, pp. 64–71, 2012.

[21] A. Barreto, P. Costa, E. Yano, *Using a Semantic Approach to Cyber Impact Assessment*, 8th International Conference on Semantic Technologies for Intelligence, pp. 101–108, 2013.

[22] G. Jakobson, *Mission Cyber Security Situation Assessment using Impact Dependency Graphs*, In Fourteenth International Conference on Information Fusion, IEEE, 2011, pp. 1–8.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 2014.

[24] M. Henrion, *Practical Issues in Constructing a Bayes' Belief Network*, In Third Conference on Uncertainty in Artificial Intelligence, 1987.

[25] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd Edition, Cambridge University Press, New York, NY, USA, 2009.

[26] A. Motzek, R. Möller, M. Lange, S. Dubus, *Probabilistic Mission Impact Assessment based on Widespread Local Events*, NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, June 2015.

[27] M. Bastian, S. Heymann, M. Jacomy, *Gephi: An open source software for exploring and manipulating networks*, In International AAAI Conference on Weblogs and Social Media, 2009.

[28] L. D. Raedt, A. Kimmig, H. Toivonen, *ProbLog: A Probabilistic Prolog and Its Application in Link Discovery*, In IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 2462–2467.

G SEMANTIC NORMALIZATION AND MERGING OF BUSINESS DEPENDENCY MODELS

Notes *This paper will be published at CBI 2016: 18th IEEE Conference on Business Informatics in Paris (France) in between the August 29th and September 1st in 2016.*

Abstract

Assessing potential threats and impacts relevant for a company, requires a detailed analysis of a company's business processes and functions down to a level of infrastructure resources, in the form of one business dependency model. Required information is frequently encapsulated in BPMN models per process, but pose an eminent problem of fusing and merging multiple sources into one model. Experts defining BPMN models possibly use different nomenclature, descriptions, and references towards common entities, leading to semantically overlapping partial dependency models. Merging multiple partial dependency models is a novel problem related to the business process matching problem, but origins from an orthogonal perspective. In this paper we propose a business dependency model normalization and matching approach by exploiting structures and dependencies of business resources, which neither requires linguistic processing nor "fuzzy" matching processes.

G.1 Introduction

Understanding and assessing potential impacts relevant for a company or mission is a pertinacious problem and a novel research area. For example, a local impact caused by an event on a distant node, say, user workstation, might lead to a causal chain of operational failures, leading to an impact on a company, as some critical devices where affected required in critical business processes. In order to understand which resources of a company are critical, and to what extent they represent criticality, various approaches, for example, by Jakobson [Jak11] or Motzek [MMLD15] employ a dependency analysis of a company and decompose a company into its business processes, depending on tasks requiring information from data stores. As an example, assume that it must be understood which processes are critical to a company and which (groups of) devices are required *directly* during all business processes. It is a common practice, e.g., proposed by Barreto [dBBdCY13], Musman [MTT+10] or Motzek [MMLD15] to extract such information from BPMN models by an extraction of involved entities in each modeled process. However, when processing multiple models a common problem becomes eminent: Multiple experts defining BPMN models use different nomenclatures, descriptions, languages and references for common entities. Therefore, a naive dependency model extraction from multiple BPMN models leads to duplicate entities leading to incorrect impact assessments.

The business process matching problem is well investigated, for example by Dijkman [DDG09], but solutions, as we outline throughout this article, are not directly applicable towards business dependency model matching. Moreover, approaches required to match dependency model are also related towards ontology matching (cf. [SE13]), which, however, frequently rely on natural language processing. In the

scope of this paper we position and relate problems around business dependency models between "classical" business process matching, ontology matching and graph isomorphism problems. We identify two problems, leading to an assumption that existing approaches are not directly applicable: In a real world use case, descriptions and identifiers of entities were sounding so similarly, s.t., any linguistic matching is bound to fail. Further, the business dependency model matching problem is less restrict than the business process matching problem. In order to emphasize the latter, we present the following example: Two modeled business processes, e.g., two BPMN models, do not match exactly, but a dependency analysis still leads to an exactly matching model. For example, a different ordering of requests, e.g., to a webserver and a database, is likely to represent two different (but, partially matching) business processes, but does not lead to an exact match. However, both processes are dependent on the web- and database server to the same extent, and both dependency models will match exactly. We say that a dependency analysis is performed orthogonally to a processing analysis. Nevertheless, we believe that in most cases an ordering of tasks in a business process is only marginally relevant to solve the (partial) business process matching problem and that our decomposition approach can help to solving the business process matching problem as well. As discussed by, e.g., Dijkman [DDG09] or Shvaiko [SE13], both, process matching and ontology matching, are known to be very hard problems. In this article we present a dependency-model matching and merging approach that scales linearly with the number of to-be-processed dependency models and number of nodes in a model.

The contribution of this paper can be summarized as follows: This paper discusses matching and merging problems associated with business dependency models and relates emerging problems towards similar problems among business process models. Without introduction of "fuzzy" or approximate matching techniques, we present an approach that solves the exact business dependency model matching problem and scales well with the number of to be matched models and that provides a working base for future partially matching approaches. We believe that the presented approach is applicable towards the (partial) business process matching problem as well.

The remainder of this paper is structured as follows: In Sec. G.2 we introduce a business dependency model and an associated use case. We formalize the business dependency model matching and merging problem in Sec. G.3 and present an approach to solve the matching and merging problem. We experimentally evaluate and discuss our approach in Sec. G.4 and conclude in Sec. G.5.

G.2 Business Dependency Models

An impact assessment is used to address potential impacts onto a higher goal, from widespread events which lead to local impacts. For example, a local impact caused by an event on a distant node, might lead to a causal chain of operational failures, leading to an impact on a company. Understanding these impacts is a pertinacious problem and current work uses adhoc solutions based on handcrafted algorithms. While such approaches deliver early results, their assessments need to be verified and validated by large amounts of data—which is not always available.

Motzek et al. introduce an approach towards impact assessment based on a probabilistic graphical model in [MMLD15], which defines a well-understood problem, namely, probabilistic inference in graphical models, on which an assessment can be reduced. By resorting to a probabilistic model, the use of conditional probability distributions allows for a local view on assessments, without a need to understand a specific use case nor any algorithmic properties. It is this local view, which allows a validation of defined data. This means that assessments from experts can be used directly without global normalization factors and experts are not forced outside their expertise. By the use of multiple probabilistic models,

expertise from different experts, namely, business, operational and security experts, are combineable in one large probabilistic graphical model for mission impact assessment. For example, an expert in charge of assessing a business dependency analysis must not reason about operational dependencies of, say, a critical webpage on a complex computation cluster and data warehouse. Likewise, an operational expert assessing dependencies inside computational clusters, must not reason about which devices are directly critical and which contribute passively to critical devices—a distinction often not uniquely identifiable.

However, Motzek et a. [MMLD15] heavily base their approach on an expert's design of a dependency model for a company. An expert is not always available or, maybe, does not even exist. Still, as they argue as well, such information is likely to be already present in the form of BPMN models, as also suggested by, e.g., Barreto [dBBdCY13]. It is straightforward from the definition of their business (or mission) dependency models on how such an extraction is performed. Motzek et al. [MMLD15] extend a model by Jakobson [Jak11] and model mission dependencies as shown in Figure G.2 as a graph of *mission nodes* (MN). A *company* is dependent on its *business processes*. A business process is dependent on one or more *business functions*, which are provided by *Business resources*. Figure G.2 shows a dependency graph of business relevant objects for a small company consisting of two business processes, requiring a total of four functions provided by four resources. Relations to an original BPMN model for BP_1 (refer Fig. G.1) are straightforward, where each BPMN model represents one business process, a BPMN-task represents a business function and BPMN-data stores (and references) represent business devices. In

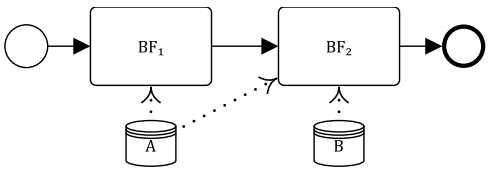


Figure G.1: Example BPMN 2.0 model sketch for the BP_1 business process shown in the dependency model of Figure G.2.

the fact, the mission dependency model represents a probabilistic graphical model and degrees of dependencies are modeled as local conditional probability distributions. An ordering and sequence of nodes in a BPMN model is exploited towards obtaining assessments of these distributions. For example, a parallel path can provide information that a redundancy in tasks and devices is present. We argue that degrees of dependencies are not relevant for the business dependency model matching problem.

For the scope of this paper we formally define a business dependency model from a graph perspective.

Definition 23 (Business dependency model)**.** *A business dependency model (MG) is a directed acyclic graph (DAG) as a pair $\langle \vec{V}, \vec{E} \rangle$ of vertices V and edges E. Vertices $\vec{V}$ are categorized according to their semantic as business- devices ($\vec{BD}$), -functions ($\vec{BF}$), -processes ($\vec{BP}$), and -company (BC). For the scope of this work we consider that a business dependency model is created for a single BC. The ordering $BD \prec BF \prec BP \prec BC$ represents the topological ordering of graph MG. Every edge $E \in \vec{E}$ represents a dependency. Each vertex $V \in \vec{V}$ is associated with attributes ID, $Name$, $Description$, for which we write V^a when referring to an attribute a of V. Let $V \in \vec{V}$ and let $\vec{E}_v \in \vec{E}$ be the set of edges directed to V, then let $\vec{d}_V$ be the set of vertices from which $\vec{E}_v$ origin, i.e., $\vec{E}_v$ is the set of dependencies of V. Every vertex $V \in \vec{V}$ a conditional probability distribution $P(V|\vec{d}_V)$ is given.* ▲

In fact, as outlined in more detail in Sec. G.4, we perform impact assessment as described by Motzek [MMLD15] in a real world use case for a large industrial company. Although, information

was not given in the form of BPMN models, business processes were described to us by multiple experts by word of mouth. Fusing all information into one model required to match different descriptions to the same entities. In particular, some experts used host names to describe ICT resources, where others used IPs. Further, sometimes business functions where named arbitrarily as an abstract conglomeration of resources, whereas another expert described valuable information to be included in the description of a business function and a business process. We believe that these problems are highly likely to be evident as well, if experts create BPMN models and an automatic extraction is performed.

As outlined before, matching business processes is not sufficient for matching dependency models, which we formally discuss in the following section and outline and implement possible solutions.

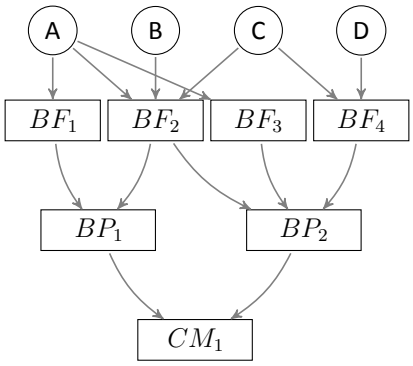


Figure G.2: A mission dependency model representing a small company CM_1 with two identified critical business processes, transitively dependent on critical resources A, B, C and D. Conditional probability distributions ommitted. If BP_1 and BP_2 are given separately, then BF_2, A, B and C are present multiple times with possibly ambigue identifiers and must be matched and merged.

G.3 Merging and Matching Problems

A business dependency model is required to obtain a view onto a company to assess potential impacts. If a dependency model contains multiple entities twice, an impact is accounted for multiple times during an assessment leading to spurious results. As sketched in the previous section, a transformation from a BPMN model towards a dependency model is straightforward by considering solely the set of modeled BPMN-tasks without their sequential orderings. However, once multiple BPMN models, i.e., multiple business processes, are processed, sets of data stores (business devices) and tasks (business functions) are likely to overlap semantically and must be merged towards one unified model.

Linguistic approaches, such as string matching algorithms to identify common entities by their description, name or ID however where bound to fail in our use case. For example, in our use case, business functions, were labeled “BF-HVD”, “BF-HVC”, “BF-MVD” and “BF-MVC”, which represented completely different functions, but involved partially matching references to business devices. While in a classic BPMN modeling approach, the sequential ordering and structure of tasks delivers sufficient information to exclude false positive linguistic matches, the softening of structure in dependency models does not allow a linguistic matching anymore.

We therefore present an approach towards solving a business dependency model matching problem based on a pure structural analysis of the dependency model and an existing inventory of addressable resources. In the following we carefully differentiate between multiple matching and merging problems

in order to clearly derive a definition of when two models "match" and how a "merging" should be executed. We present an approach for solving a specialized problem and experimentally evaluate that it solves the semantic normalization and merging problem.

Definition 24 (Semantic normalization and merging problem, SMP)**.** *Let m_1, m_2 be mission dependency models. Then, the semantic business dependency model normalization and merging problem (SMP) is the problem to obtain m from m_1 and m_2 without a duplication of nodes and vertices that bear an identical semantic and to normalize all vertices in m into a standardized format.* ▲

Informally, a solution to the SMP shall represent how a human would merge and normalize two presented dependency models. Essentially, the SMP is only solvable directly by an expert, because a clear definition of "identical semantic" is missing. To obtain a more clear definition, we define the following.

Definition 25 (Business dependency model matching problem, BDMP)**.** *Let m_1, m_2 be mission dependency models and let mn be a vertex of an arbitrary mission dependency model. Then the BDMP is the problem to find the set $\vec{mn}$, s.t., for all $mn \in \vec{mn}$ holds $\forall mn \in \vec{mn} : \exists (mn_1 = mn) \in m_1 \wedge (mn_2 = mn) \in m_2$ with respect to an equality operator $=$.* ▲

The simplest definition of an equality operator, would be an exact equality, for which we write $=_*$. Namely for $=_*$ to hold, all attributes and dependencies of two nodes must exactly match. As argued before, we doubt that, due to multiple experts, designers and naming schemes, a matching problem based on $=_*$ is of any avail. Still, it is indisputable that two dependency models that match exactly ($=_*$) are seen as semantically identical by a human as well. Further, we assume that if two nodes share a unique ID, i.e., share a non-trivial (randomly) generated unique identifier, written $=_{id}$, one usually assumes semantic equality as well. This means that if an expert trusts that IDs are unique, an experts performs a *semantic* matching on IDs. We will therefore develop an approach that firstly normalizes all nodes to unique IDs, s.t., the BDMP w.r.t. $=_{id}$ does represent a semantic match as performed by an expert, i.e., represents the SMP.

Definition 26 (Identity equality)**.** *Let mn_1, mn_2 be two arbitrary mission nodes. Let $\vec{mn}$ represent the set of* all *possible $\vec{mn}$ existing. Then mn_1 and mn_2 are identity equal, written $mn_1 =_{id} mn_2$, iff, $mn_1^{id} = mn_2^{id}$ and $\nexists mn_3 \in \{\vec{mn} \backslash mn_1, mn_2\} : mn_3^{id} = mn_1^{id}$. Two sets of mission nodes $\vec{mn}_1$, $\vec{mn}_2$ are identity equal, if every node in $\vec{mn}_1$ is identity equal to exactly one node in $\vec{mn}_2$ and vice versa.* ▲

We use the definition of identity equality in order to reduce the SMP problem to a more tangible problem: If the BDMP for $=_{id}$ represents the SMP, then the problem that needs to be solved is a normalization, s.t., we uniquely identify individual entities. This means that not complete graphs must be matched, but only single entities. As demonstrated later, this significantly reduces computational complexity.

To normalize all nodes to be identified by unique IDs, we exploit that data stores, i.e., business devices are not only referenced by business models, but are part of an infrastructure and are referenced by, e.g., large inventories, i.e., other independent sources. While in ICT related use cases such inventories can be created automatically by network mappers and scanners, in other use cases, e.g., in logistics, such inventories are present as well in the form of stock lists or equipment lists. We therefore obtain identity equality by normalizing all business devices based on inventory equality.

Definition 27 (Inventory equality)**.** *Let bd_1, bd_2 be business devices and let I be an inventory. Then bd_1 and bd_2 are inventory equal w.r.t. I, written $bd_1 =_I bd_2$, if an non-empty arbitrary attribute of bd_1 is equal to an arbitrary non-empty attribute of exactly one resource $g \in I$ and an arbitrary non-empty attribute of bd_2 is equal to an arbitrary non-empty attribute of g.* ▲

For the scope of this article and our use case, we use attributes representing IP-addresses (IPv4 and 6), MAC-addresses, hostnames and unique node identifier. During conversations with experts from our use case we found that devices were most often referenced by their hostname and (static) IP address. Extensions to inventory equality, for example, involve a partial matching of hostnames or to some extent a match of IP ranges.

As mentioned before, such inventories only exist for physically existing devices and do not cover modeled tasks or processes themselves. Still, as one is now able to uniquely identify dependencies of tasks (business functions) onto business devices, we argue that two functions depending on the same set of devices must be semantically identically as well. We therefore define structural equality.

Definition 28 (Structural equality)**.** *Let mn_1, mn_2 be mission nodes. Let $\vec{d}_{mn}$ represent the set of dependencies of a mission node mn. Then mn_1 and mn_2 are structurally equal, written $mn_1 =_s mn_2$, iff, $\vec{d}_{mn_1} =_{id} \vec{d}_{mn_2}$ and $\vec{d}_{mn_1} \neq \emptyset$.* ▲

For the scope of this work we only consider an exact structural equality, i.e., both dependencies sets must match completely. It is straightforward to extend this work to apply partial structural matching by defining a difference metric between both sets $\vec{d}_{mn_1}$ and $\vec{d}_{mn_2}$ and allowing a degree of deviation. A modification of our approach towards partial matching is later given as a remark on future work. Still, once a degree of deviation is allowed in structural matching, an extent of "fuzzyness" is introduced which might lead to false positives. The following remark sketches a combination of attribute and structural matching to prevent false positive partial structural matching.

Remark 6 (Structural equality linguistic sufficiency conditions)**.** *So far attributes are not considered, and, attributes are deemed to be equivalent given a structural match. Still, there is the possibility that semantically different business functions or processes, in fact, produce the same dependency model, or are almost identical, if one allows for a degree of deviation. In such a particular situation, a structural equality will lead to a false-positive merging, but could be prevented by using attributes as sufficient conditions. For example, two business functions, named "Credit Card Fraud Detection" and "Customer Balance", will both directly dependent on a customer database and a transaction history. If a degree of deviation in dependencies is allowed, it is likely that both will be structurally identical. A simple linguistic comparison can prevent a false positive match in this case. Due to the assumption that structural equality already a high degree of (or even a a complete) semantic similarity is given, a linguistic comparison must only be performed as a false-positive prevention on the set of structurally equivalent nodes, and can be less restrictive. If a linguistic comparison would be employed as a first-step classification, a function "Credit Card Fraud Detection" would not be matched with an identical, but named "CCFD" in a different business process.*

We therefore envision to use attributes solely as a plausibility check. To do so, we group structural identical nodes (Struct$_i$), as shown in Figure G.3 as a sorted collect of structurally identical nodes inside them, which again are grouped by leading attributes that shall be used for a linguistic comparison. This is sketched as an example for a leading attribute of the name in Figure G.3. This has the benefit that post a structural matching, a plausibility linguistic matching can be performed efficiently and can be extremely unrestrictive. For example, a descriptive attribute can be seen as being linguistically identical, once a certain degree of non-trivial words are found. ▲

In summary, we obtain a clear definition of when ($=_{id}$) and how (exploiting $=_I$, $=_s$) models are seen as semantically equivalent. Using these definitions, we propose an algorithm as presented in Alg. 3

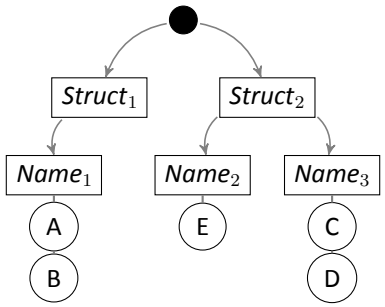


Figure G.3: Structural model to perform a postponed linguistic sufficiency check on structurally identical nodes. $A - E$ represent business functions (or, respectively, processes), grouped per structural identity (*Struct*$_i$). Structurally equivalent nodes are grouped by (partially) matching leading attributes used for linguistic matching purposes; in this example *Name*.

that firstly enforces identity equality on business devices through exploitation of inventory equality, and consecutively creates identity equality on business functions by exploitation of structural equality $=_s$. Respectively, identity equality is obtained among business processes through exploitation of structural equality on business functions.

Algorithm 3 Incremental match and merge

1: **procedure** MATCH-AND-MERGE(m_1 to m, given I)
2: **for all** $BP \in (\vec{BP} \in m_1)$ **do**
3: **for all** $BF \in (\vec{BF} \in m_1)$ **do**
4: **for all** $BD \in (\vec{BD} \in m_1)$ **do**
5: $BD \leftarrow BD_{norm} : BD_{norm} =_I BD$
6: $m \leftarrow m \cap BD$
7: $BF \leftarrow BF_{norm} : \{(BF_{norm} \in m) =_s BF\} \vee BF$
8: $m \leftarrow m \cap BF$
9: $BP \leftarrow BP_{norm} : \{(BP_{norm} \in m) =_s BF\} \vee BP$
10: $m \leftarrow m \cap BP$
11: **return** m

Informally, the algorithm firstly searches and normalizes each business device in a supplied inventory I (depending on the definition of $=_I$ a partial match is defined as well). Based on the normalized devices, which are added to the target model m, structural matches of business functions in m_1 according to $=_s$ are searched in m. Unmatchable business functions are appended to the target model m and business processes of m_1 are normalized and matched respectively. For efficiently finding a structural matching entity (Lines 7 and 9) we use Alg. 4 which encodes a dependency structure as a unique string representation. Alg. 4 creates structure-identifying strings, which are natively sorted by increasing numbers of dependencies. That all reference strings are sorted ascending by increasing size of dependencies is achieved prefixes generated in Line 4. If fixed-length identifiers for dependencies are used, a structurally equivalent mission node can be found by a binary search in the set of reference strings.

Example 8 (Structural string encoding)**.** *To demonstrate the structural encoding identifier, we refer to Fig. G.2. A structural encoding for BF_1 is 1-A and for BF_2 is 3-ABC. Then, a structural encod-*

Algorithm 4 Dependency string encoding

1: **procedure** ENCODE-STRUCTURE(mission node mn)
2: **if** $mn \in \vec{BD}$ **then return** mn^{ID}
3: $\vec{d}_{mn} \leftarrow sort(\vec{d}_{mn})$ by $|\vec{d}_d|, d \in \vec{d}_{mn}$
4: $C \leftarrow |\vec{d}_{mn}|-$
5: **for all** $d \in \vec{d}_{mn}$ **do**
6: $C \leftarrow C \cap$ ENCODE-STRUCTURE(d)
7: **return** C

ing of BP_1 is `2- 1-A 3-ABC` (white-space are added for better readability). Likewise, one obtains `3- 1-A 2-DC 3-ABC` for BP_2

Note that if no fixed-length identifiers are used, further delimiters are required, e.g., for BP_1 like `2-(1-(A)).(3-(A).(B).(C))`. ♦

Given a dependency model m consisting of n_{BP} business processes, n_{BF} referenced functions and an inventory consisting of n_{BD} devices, to which a dependency model m_1 with n^*_{BF} and n^*_{BD} is to be merged, the algorithm performs a structural match in in $n^*_{BD} \cdot \log n_{BD} + n^*_{BF} \cdot \log n_{BF} + \log n_{BP}$, if one uses a binary search to find matching devices and dependency structures. As we argue that an inventory equivalence and structural equivalence must represent a semantic equivalence, one can assign unique IDs and therefore obtains a solution for the SMP.

Theorem 2. The procedure represented by Alg. 3 solves the BDMP with respect to a semantic equivalence. ▲

This theorem is proven in combination with an evaluation of the complete procedure including a merging operation in the following section.

The structural encoding can be extended towards partial matching of business process dependencies, which we outline in the following remark.

Remark 7 (Partial dependency matching)**.** *We exploit structural equivalence, i.e., we exploit that two entities that are dependent on the same dependencies are seen as identical. If, for example, nine out of ten dependencies match, one could argue that both entities are still seen as identical. The encoding to reference structures used by Alg. 3 bears significant advantages for such a partial structural match, as it can be directly seen as a string alignment problem. Say, one is given two string-encodings for business processes by `CD ABC DEF` and `Q ABC CDE DEG` (excluding the dependency size prefixes). An approach to align both dependency structures is fairly similar to computing a Damerau-Levenshtein-Distance between both. We envision the usual four modification patterns as $i$nsertion, $d$eletion, $s$ubstitution and $t$ransposition, where transposition is only allowed on complete business-function blocks (delimited by white-spaces) but is cost-free. Then, the first dependency structure is aligned to the second by substituting `F` and `G` in `DEF`, adding `Q`, transposing `CD` by one, and adding `E` to `CD` with an edit-score of 3 (transpose is free).*

Notwithstanding, a partial alignment can also be performed on a deeper encoding level, i.e., for matching business functions directly. However, then information about partial matches is eliminated too early. It is beneficial to perform this alignment on business processes, as one gains further information: If one

merges CDE and CD too early, i.e., during matching of business function, one loses the information that, in fact, CD and CDE are used by two higher-level partially matching business processes. For example, if two functions CDE and CD are used by two to-be-matched business processes which are otherwise completely different, it is less likely that functions CDE and CD match. In contrary, if both functions are used by otherwise completely identical business processes, it is more likely that the functions match. By performing an alignment at a business processes level, one preserves and can incorporate this knowledge. ▲

As usual, given a matching set between two models, it is the common problem to unify all information towards one model.

Definition 29 (Business dependency model merging problem, BDFP)**.** *Let m_1, m_2 be business dependency models and let $\vec{mn}_=$ be a solution to the BDMP between m_1,m_2 w.r.t. $=$. Then the BDFP is the problem to transfer all attributes of pairs $\langle mn_1, mn_2\rangle \in \vec{mn}_=$, where $mn_1 = mn_2$ into a unified node mn_n.* ▲

Essentially, if all matching nodes are not exactly equivalent, lossy and lossless merging operators exist. As business devices are normalized through inventory equality, business devices immediately are exactly equivalent and no merging operation needs to be executed. Still, merging different descriptions and names of, e.g., business functions remains as a problem. Nevertheless, to perform an impact assessment merged descriptions and names are not required. We therefore, for the scope of this article, use a simple lossless merging approach.

Proposition 1 (Lossless business dependency model merging)**.** Given a BDFP for models m_1, m_2 and $\vec{mn} =$, a lossless exact solution is to transfer each $\langle mn_1, mn_2\rangle \in \vec{mn}_=$ into mn_n by concatenating all attributes. ▲

Note that, as we use an exact structural match, only attributes need to be merged and that dependencies are per definition equivalent.

In summary, our approach performs a structural match between two dependency models and scales linearly with the number of to be matched and merged dependency models. A pure merge based on structural equivalence is possible, as ground truth is incorporated by consulting external sources where individual assets of modeled processes are referenced as well. In the following section we experimentally evaluate that our approach based on a modeled derived from a real world use case which is artificially duplicated and heavily modified.

G.4 Experiments and Discussion

We implement and apply a probabilistic mission impact assessment as presented by [MMLD15] in a large industrial enterprise. In this use case a complete business dependency model was created by hand by us originating from different descriptions of experts from the company. We obtained four business processes, requiring seven business functions provided by 26 business devices, where all dependencies are deeply meshed, as a high degree of redundancy is present in the industrial network. Effectively, large sets of business functions and associated devices overlapped in the requirements of each business process. In order to test our approach we extract a partial dependency model for every business process from our model and artificially manipulate it, preserving structures, but manipulate almost all semantic information. By doing so, we simulate common errors that would occur when extracting partial dependency models from individual BPMN models. Attributes of business devices are

only modified to an extend that they are still identifiable by $=_I$. To be precise, we randomly remove unique identifiers and manipulate, switch and remove description- and name- attributes, and randomly add words, letters and punctuation. In this evaluation, the number of total business processes remains constant. Results, shown in Figure G.4, show that the total running time scales linear with the total amount of processed dependency model, i.e., the time for to perform a match and merge is constant.

To show that our approach scales linear with the number of merged business processes, we randomly generate $10\,000$ business processes by random combinations of business functions. Every generated business processes is modified ten times by the aforementioned process. Therefore, we match and merge $100\,000$ partial dependency models representing $10\,000$ unique processes and denote the total running time over the number of acquired unique business processes in Fig. G.5, i.e., one obtains 10 measurements for every business process. As evident from Fig. G.5, the approach scales linearly with the number of unique business processes, i.e., performs a match and merge in constant computation time. We were unable to measure a significant increase in computation time required for a match based on the number of processed models. In fact, we were able to process $100\,000$ partial dependency models in less than seven minutes on a standard Intel Core i5-4300U mobile CPU.

After all evaluations, i.e., after processing $120\,000$ models, the obtained merged dependency model included the same business processes (including the artificially added), the same business functions and the same business devices as the original model and the generated graph is isomorph to the original model, i.e., all dependency structures were preserved and no entities were removed or duplicated. Therefore, the presented approach based on structural and inventory equivalence indeed solves the SMP for the generated sets, as stated by Thm. 2 ■. Admittedly, this evaluation is also targeted to evaluate the complexity of our approach and the manipulation is chosen in a way such that inventory equality must be assured by our approach. Still, all generated dependency models are manipulated to an extent, where any linguistic matching would fail and resemble commonly made errors. In fact, a structural match is the only left matching ability, even for an expert in our generated test set. Furthermore, considering that (partial) business process matching is a very hard problem ([DDG09]) and that our approach scales linearly is a promising result. It is likely that two matching dependency models are based on two (partially) matching business process models and can, as well, be used as a preprocessing step for business process matching.

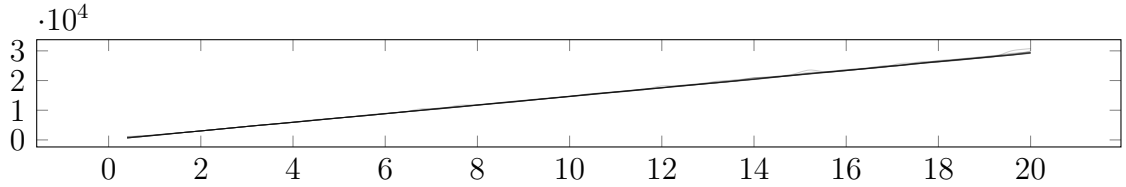


Figure G.4: Total computation time (t_{mm}, ordinate in seconds) for incrementally matching and merging is linear within the number of dependency models n_{MG} (abscissa) merged so far. 13 evaluations over a set of $20\,000$ models displayed superimposed.

We motivate a view of business process in the form of business dependency models to perform an impact assessment, where dependency models are a required input data. Viewing companies, or most often missions, as dependency models decomposing an asset into multiple layers is as well done by Barreto et al. [dBBdCY13] who introduce a well-understood modeling technique and also use BPMN models to acquire knowledge, but do neither focus nor discuss matching and merging gathered information.

Albanese et al. present in [AJJP13] a well-modeled formalism for complex inter-dependencies of missions as a set of tasks, but do not consider the acquisition of such knowledge. Further, their models are

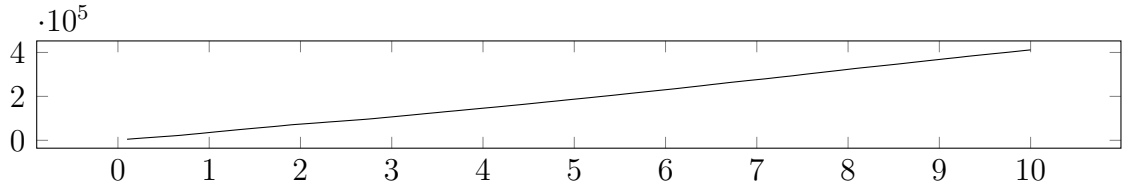


Figure G.5: Total computation time (t_{mm} ordinate in seconds) for incrementally matching and merging is linear within the number of unique business processes n_{BP} (abscissa) obtained so far. This means that the computation time to perform a match and merge is constant even for increasing sizes of acquired models. Mean of 10 measurements displayed.

tree-structured, i.e., accept duplication of multiple entities. Jacobson [Jak11] presents a well understood conceptual framework using interdependencies based on operational capacity, his work is extended by Motzek et al. [MMLD15] who both view compies as a dependency graph. While Motzek et al. do discuss an acquisition of knowledge, Jacobson does not. Both do not consider an automatic extraction, matching and fusing of multiple information sources.

Goodall et al. [GDK09] and D'Amico et al. in [DBGW10] discuss mission modeling with the use of ontology based descriptions. Especially, Goodall et al. discuss fusion and data integration of multiple sources. However, as their work is based on a different conceptual level, the use of ontologies is not applicable to our intended use case of mission impact assessment. Still, the range of ontology merging and matching is a deep research area as well, frequently relying on natural language processing, which is not exploitable in the first place for our work. Being a broad field or research we refer to overlooking work by Shvaiko and Euzenat [SE13] and [SE08].

G.5 Conclusion

We introduce, discuss and formalize novel problems emerging around business dependency models, which are related to business process models, but are viewed from a orthogonal perspective. Where business process models intend to cover sequences of actions, dependency models try to grasp an overview of a company as a conglomerate of abstract, multiple concepts down to a deeper layer of resources. A view is required in order to understand possible impacts and threats to a company based on deeply meshed resource infrastructures, e.g., found in large industrial ICT and ICS focused companies.

We present and discuss variations of an algorithm able to fuse and merge multiple sources for business dependency models into one consistent model on which an impact assessment can be performed. We demonstrate that the approach provides valuable starting points for extended work and show that the approach scales well. Future work is dedicated to integrate partial dependency model matching and dedicated to research on the applicability of dependency model matching towards business process model matching. Moreover, we investigate more deeply on partial structural matching including linguistic plausibility checks. By considering allocations of larger partially-matching linguistic subgroups will allow for refining matched groups at higher levels.

Bibliography

[AJJP13] Massimiliano Albanese, Sushil Jajodia, Ravi Jhawar, and Vincenzo Piuri. Reliable Mission Deployment in Vulnerable Distributed Systems. In *DSN Workshops 2013: 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop, Budapest, Hungary, June 24-27, 2013*, pages 1–8. IEEE, 2013.

[dBBdCY13] Alexandre de Barros Barreto, Paulo Cesar G. da Costa, and Edgar Toshiro Yano. Using a Semantic Approach to Cyber Impact Assessment. In *8th Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax VA, USA, November 12-15, 2013*, pages 101–108, 2013.

[DBGW10] Anita D'Amico, Laurin Buchanan, John Goodall, and Paul Walczak. Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions, and Users. In *5th International Conference on Information Warfare and Security*, pages 8–9, 2010.

[DDG09] Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *BPM 2009: 7th International Conference on Business Process Management, Ulm, Germany, September 8-10, 2009*, pages 48–63, 2009.

[GDK09] John R Goodall, Anita D'Amico, and Jason K Kopylec. Camus: Automatically Mapping Cyber Assets to Missions and Users. In *Military Communications Conference*, pages 1–7. IEEE, 2009.

[Jak11] Gabriel Jakobson. Mission Cyber Security Situation Assessment using Impact Dependency Graphs. In *FUSION 2011: 14th International Conference on Information Fusion, Chicago, Illinois, USA, July 5-8, 2011*, pages 1–8, 2011.

[MMLD15] Alexander Motzek, Ralf Möller, Mona Lange, and Samuel Dubus. Probabilistic Mission Impact Assessment based on Widespread Local Events. In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, pages 16–22, 2015.

[MTT+10] Scott Musman, Aaron Temin, Mike Tanner, Dick Fox, and Brian Pridemore. Evaluating the Impact of Cyber Attacks on Missions. In *5th International Conference on Information Warfare and Security*, pages 446–456, 2010.

[SE08] Pavel Shvaiko and Jérôme Euzenat. Ten Challenges for Ontology Matching. In *OTM 2008: On the Move to Meaningful Internet Systems, Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Part II*, pages 1164–1182, 2008.

[SE13] Pavel Shvaiko and Jérôme Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1):158–176, 2013.

H INDIRECT CAUSES IN DYNAMIC BAYESIAN NETWORKS REVISITED

Notes *This article has been presented and published at IJCAI 2015: 24th International Joint Conference on Artificial Intelligence in Buenos Aires (Argentina) from the 25th to 31th of July in 2015.*

Abstract

Modeling causal dependencies often demands cycles at a coarse-grained temporal scale. If Bayesian networks are to be used for modeling uncertainties, cycles are eliminated with dynamic Bayesian networks, spreading indirect dependencies over time and enforcing an infinitesimal resolution of time. Without a "causal design," i.e., without anticipating indirect influences appropriately in time, we argue that such networks return spurious results. By introducing activator random variables, we propose template fragments for modeling dynamic Bayesian networks under a causal use of time, anticipating indirect influences on a solid mathematical basis, obeying the laws of Bayesian networks.

H.1 Introduction

Dynamic Bayesian networks (DBNs) are an extension to Bayesian networks motivated from two perspectives, on the one hand as a manifestation of cyclic dependencies over time, closely related to Markov models [Mur02], on the other hand as a stationary process repeated over time in fixed time slices [GK95]. Considering [PR03] who emphasized that Bayesian networks should be a direct representation of the world instead of a reasoning process, both views seem to be conflicting. A stationary model repeated over time with cyclic dependencies would expand to infinity already for one timeslice. Therefore, cyclic dependencies in a stationary process are restricted [Jae01] and forced into a strict order, e.g., state variables of time t are only dependent of states at $t-1$. Unfortunately, this means that evidence at a certain time point does not affect states at this time point, but one slice later.

In the extreme form of a DBN, every state variable is dependent on every other. In that case, there is no option to leave such dependencies in their causally correct same timestep as every dependency would cause cycles. Therefore, states can only be dependent on states from a previous timestep. However, this poses conflicts in causality, as a) the temporal causality is simply inaccurate and b) no indirect effects are considered, enforcing an infinitesimal resolution of time instead of a world-representing designed time and heavily limits the usage of a DBN.

To circumvent this problem, basically two options are available. As investigated by [BFGK96] variables might be independent in certain contexts, which would allow a causally correct network generation from rules such as those presented in [GK95] or [NH97]. Still, then rules would need to be designed with a procedural view, degrading a BN to a procedural tool in a reasoning process, rather than designing it as a first-class declarative representation. Further, such rules would inherently be cyclic and might cause problems as stated by [NHH95]. A second option would be to heavily restrict a DBN to specialized observation sets, e.g., to "single observations at a time" as done in [SDW05], s.t. no indirect causes need to be considered. Again, this implies that observations are made at an infinitesimal resolution of time.

The contribution of this paper can be summarized as follows. By introducing activator random variables, we propose template fragments for modeling DBNs under an unrestricted use of time, anticipating indirect influences on a solid mathematical basis, obeying the laws of Bayesian networks. This is beneficial for application contexts where causal models arise naturally and require a view over time, e.g., automatic learning of causal influences from coarse observation sets and—as a long-term goal—finding causally correct explanations and relations in (temporally uncertain) knowledge bases requiring anticipation of causal chains, e.g., DeepQA [FBCC+10] or the Knowledge Vault [DGH+14].

We discuss preliminaries on DBNs and context-specific independencies as introduced by [BFGK96] and [HHNK95] in Sec. H.2. By extending DBNs with activator random variables, we propose Activator Dynamic Bayesian Networks (ADBNs) in Sec. H.3, derive common operations on ADBNs such as filtering and smoothing in Sec. H.4, discuss our results in Sec. H.5, and conclude with Sec. H.6.

H.2 Dynamic Bayesian Networks: Preliminaries

A DBN models a stationary Markov process of state influences and transitions that is repeated over time.

Notation 1 (State Variables). *Let X_i^t be the random variable of the i^{th} state variable X_i at time t, where X_i^t is assignable to one of its possible values $x_i \in \text{dom}(X_i^t)$. Let $\vec{X}^t$ be the set of all n state variables at time t, s.t.,*

$$\vec{X}^t = \left(X_1^t, \ldots, X_n^t\right)^\intercal.$$

Let $P(X_i^t = x_i)$ (or $P(x_i^t)$ for brevity) denote the probability of state X_i having x_i as a value at time t. If $\text{dom}(X) = \{true, false\}$ we write x^t for the event $X^t = true$ and $\neg x^t$ for $X^t = false$ as usual. If X_i^t is unspecified and not defined through a query, $P(X_i^t)$ denotes the probability distribution of X_i^t for all its possible values. ▲

Definition 30 (Dynamic Bayesian Network). *A DBN is a tuple $(B_0, B_\rightarrow)$ with B_0 defining an initial Bayesian network (BN) representing time $t = 0$ containing all states X_i^0 in $\vec{X}^0$ and a consecutively repeated Bayesian network fragment $B_\rightarrow$ defining state dependencies between X_i^s and X_j^t, with $X_i^s \in \vec{X}^s, X_j^t \in \vec{X}^t, s \leq t$. By repeating $B_\rightarrow$ for every time step $t > 0$, a DBN $(B_0, B_\rightarrow)$ is unfolded into a BN uniquely defining a joint probability $P(\vec{X}^{0:t\intercal})$. Notwithstanding, for every random variable X_i^t a local conditional probability distribution (CPD), e.g., as a CPT, is defined.*

State dependencies defined in $B_\rightarrow$ are limited, s.t. no cyclic dependencies are created during unfolding. For $t-1 \leq s \leq t$, we speak of a first order Markov property, which we want to discuss in this paper. For any probabilistic model with $t-1 \leq s < t$, i.e., states at time t are only dependent of states at time $t-1$, an acyclicity constraint in the directed graph holds. A limited set of dependencies of the form $t-1 \leq s \leq t$ are possible, as long as no directed cycles are created. ▲

Commonly, in such networks we distinguish between observable (sensors) and unobservable (hidden states) variables. For our work, we consider a fully observable Markov model containing only observable *states*.

Diagonal state dependencies (as in Fig. H.1) with $t-1 \leq s < t$ (acc. to Def. 30) are often due to syntactic constraints on (D)BNs and stand in conflict with an actual causality. Such dependencies exist causally at $s = t$, but create directed cycles. While conflicting with causality, further, dependencies on "sibling" states of one time slice are spread over the past. This means, indirect causes among siblings are not anticipated, or rather, that chain reactions are not covered.

Example 9 (Regulatory Compliance)**.** *In a company deliberately placed false information, e.g., faked payment sums for bribe money, might divulge throughout a company until every employee believes (unknowingly) in a lie. We therefore model a probabilistic regulatory-compliance checking tool using a DBN to track and query possible violations of employees over time and to track back potential sources of deliberately placed false information. If one employee believes in false information, we do not say that such an employee is "corrupt", but say that he is* credulous*. Every employee, Claire,* Don *and* Earl*, say, is represented by one state in $\vec{X}^t$. The probability $P(X_i^t)$, encodes the belief in employee X_i being credulous x_i^t or being integrous $\neg x_i^t$ at time t. We model B_0, s.t., it models our prior belief in every employee being a source of false information, i.e., B_0 is a BN containing all $\vec{X}^0$ as prior random variables; say $P(c^0) = 0.9, P(d^0) = 0.6, P(e^0) = 0.01$. Being credulous is permanent, such that $B_\rightarrow$ describes all random variables X_i^t depending on X_i^{t-1} with conditional probability $P(x_i^t|x_i^{t-1}) = 1$.*

An employee might influence another employee in his writings or, rather, in his information state. A credulous employee might therefore (undeliberately) influence his colleague such that the colleague also believes in false information, i.e., becomes credulous, too. Say, Claire influences Don, and if Claire is credulous, there is a probability of Don becoming credulous too. Further, if Don influences Earl, there is a probability that Claire influences Earl indirectly *through Don. We can model this correctly as a dependency as $C^t \rightarrow D^t \rightarrow E^t$ in $B_\rightarrow$. We assume an individual probability of 0.8 for an employee becoming credulous when being influenced by a credulous person and a noisy-or combination for every state.*

However, we want to model that all employees can influence each other, and, to assure an acyclicity constraint, we must bend the influencing dependency to a consecutive timestep in $B_\rightarrow$ (as in Fig. H.1). This is unavoidable, but is inaccurate from a world representation point of view, as indirect influences are now anticipated spuriously. Earl is now influenced by Claire through *Don from a Claire of the penultimate time. This means, a time slice must be infinitesimal small, e.g. secondly, to anticipate all indirect influences, and cannot be chosen freely to an intended use case, e.g. daily.* ♦

In our example, we now can make observations, e.g., from unheralded compliance checkups and trace a potential diffusion of false information throughout our company over time. Still, we cannot actually model an accurate representation of a world, because we have to use a modeled dimension (time) for assuring syntactic constraints of (in)dependencies.

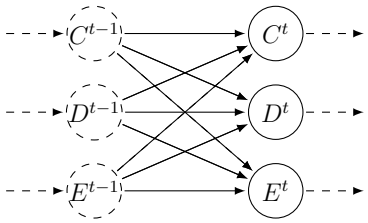


Figure H.1: A "diagonal" DBN fragment $B_\rightarrow$ for Ex. 9. E^t is only influenced indirectly by a past C^{t-2} through D^{t-1}.

Classically a conditional independency in a Bayesian network is represented by the lack of an arc between two nodes. Another kind of independencies in Bayesian networks, called context-specific independencies (CSIs), has been studied by [BFGK96] & [NH97] and has mainly been used for more efficient inference in such networks. CSIs represent dependencies in a BN that are only present in specific contexts. We extend this idea by defining special activator random variables.

Definition 31 (Activator Random Variable)**.** *We define A_{XY} to be an activator random variable which activates a dependency of random variable Y on X in a given context. Let* dom$(A_{XY}) = \{true, false\}$ *(extensions to non-boolean domains are straightforward). We define the* deactivation *criterion from a functional perspective towards the CPT as*

$$\forall x, x' \in \text{dom}(X), \forall y \in \text{dom}(Y), \forall \vec{z} \in \text{dom}(\vec{Z}) : \tag{H.1}$$
$$P(y|x, \neg a_{XY}, \vec{z}) = P(y|x', \neg a_{XY}, \vec{z}) = P(y|*, \neg a_{XY}, \vec{z}) \,,$$

where $*$ represents a wildcard and $\vec{z}$ further dependencies.

The activation *criterion describes a situation where Y becomes dependent on X, i.e., the CPT entry for y is not uniquely identified by just a_{XY} and $\vec{z}$, hence*

$$\exists x, x^* \in \text{dom}(X), \exists y \in \text{dom}(Y), \exists \vec{z} \in \text{dom}(\vec{Z}) :$$
$$P(y|x, a_{XY}, \vec{z}) \neq P(y|x^*, a_{XY}, \vec{z}) \,. \quad \blacktriangle \tag{H.2}$$

Example 10 (Activator)**.** *Claire does not constantly influence Don, but only if Claire sends a letter to Don. We can observe possible exchanges from used envelopes (possibly found in the trash bin). On such envelopes, we find multiple transfers from a coarse time interval in an imprecise or inaccurate order. For example, a transfer from Don to Earl and one from Claire to Don might include a transitive influence of Claire on Earl* at the same time*. A document transfer at time t, denoted M^t_{CD}, is then an activator for an influence of Claire C^t on Don D^t. Likewise, if we can neglect this document transfer, i.e. observe $\neg m^t_{CD}$, Don becomes independent of $Claire$ at time t.* ♦

The example shows that sometimes dependencies are modeled in $B_{\rightarrow}$ that are not always needed.

H.3 Activator Dynamic Bayesian Networks

We extend Bayesian networks such that, besides state variables, we have activators purely acting as necessary conditions for context-specific (in)dependencies.

Notation 2 (Activator Matrices)**.** *Let $A^{s\,t}$ describe the matrix of all activator random variables between time-slice s and t, s.t.,*

$$A^{s\,t} = \begin{pmatrix} A^{s\,t}_{11} & \cdots & A^{s\,t}_{1n} \\ \vdots & \ddots & \vdots \\ A^{s\,t}_{n1} & \cdots & A^{s\,t}_{nn} \end{pmatrix} .$$

Let $\vec{A}^{s\,t}_i$ denote the i^{th} column of $A^{s\,t}$ and let $\vec{\mathcal{A}}^{s\,t}$ denote the corresponding column vector of all entries of $A^{s\,t}$, s.t.

$$\vec{\mathcal{A}}^{s\,t} = \left(A^{s\,t}_{11}, \ldots, A^{s\,t}_{1n}, \ldots, A^{s\,t}_{n1}, \ldots, A^{s\,t}_{nn}\right)^{\intercal} .$$

▲

Definition 32 (Activator Dynamic Bayesian Network (ADBN))**.** *An ADBN fragment template $B'_{\rightarrow}$ consists of dependencies between states X^s_i and X^t_j, $t-1 \leq s \leq t$ (Markov-1) and matrices $A^{s\,t}$ of activators. Let $A^{s\,t}_{ij}$ be the activator random variable influencing X^t_j regarding a dependency on X^s_i, such that X^t_j's local CPT follows Eq. H.2 and Eq. H.1. Every activator is assigned a prior probability. An ADBN is then syntactically defined by $(B_0, B'_{\rightarrow})$ defining its semantic as a joint probability $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{01:tt\intercal})$.* ▲

Still, $t-1 \leq s < t$ is necessary to absolutely assure an acyclicity constraint when modeling ADBNs. Under the condition $t-1 \leq s \leq t$, possibly directed cycles are created. For our work we only consider the problematic $t-1 \leq s \leq t$ case, only containing in-time-slice dependencies (as in Fig. H.2). For brevity, we write A^t for A^{tt} excluding A^{tt}_{kk}, and, correspondingly, $\mathcal{A}^t$ and A^t_{ij}.

The constraint $t-1 \leq s < t$ for assuring acyclicity even in complete digraphs limits a causal reasoning process to direct dependencies between states.

Proposition 2 (Diagonal (A)DBN Restrictions)**.** A classic, "diagonal" (as in Fig. H.2) (A)DBN of type $t-1 \leq s < t$ is restricted in its usage to special observation sets. Indirect influences are spread over multiple timesteps and possible indirect influences inside one timestep cannot be considered. This enforces a) an infinitesimal resolution of observations, where indirect effects do not need to be anticipated or b) restricts a DBN to observations where indirect influences strictly do not occur. This implies, not a single two activators A^t_{*i} and A^t_{i*} are allowed to be probably active, i.e. the set of probably active activators must form a bipartite digraph with uniformly directed edges. Further, only up to $n^2/4$ activators are allowed to be probably active per timestep, and *all* other activators must be (i) observed to be (ii) deactive. If, in a diagonal DBN, observations can neither hold a) or b), observation- and query-(de)serializations would be needed, and $n-2$ spurious "time"-slices would need to be inserted between $t-1$ and t. In our opinion, this degrades a BN to a reasoning tool. ▲

We show, by using a modified acyclicity constraint, that in ADBNs (Fig. H.2) we can correctly anticipate indirect influences by modeling dependencies causally correctly.

Example 11 (Example continued)**.** *By extending our credulousness representing DBN with document transfer activators we obtain an ADBN with activators* $\vec{\mathcal{A}}^t = (M^t_{CD}, M^t_{DC}, M^t_{DE}, M^t_{ED}, M^t_{CE}, M^t_{EC})$ *and states* $\vec{X}^t = (C^t, D^t, E^t)$*. For every* t *we assume a prior probability for any transfer of* $P(m^t_{ij}) = 0.5$*. Still, we have to assume that every employee can influence every other, i.e., send him a document. To cover chain reactions of multiple transfers, we would then need syntactically forbidden cyclic dependencies.* ♦

The following Thm. 3 states that by using an ADBN it is indeed possible to move an acyclicity constraint from a design phase to an operations phase while maintaining a solid mathematical basis in accordance with Bayesian network semantics. This means, if the only possible mails are from Claire to Don to Earl, we could have modeled all influences correctly in one timestep during design of the DBN. Unfortunately, we do not know possible observations during design and such acyclic mail exchanges may differ in every time step.

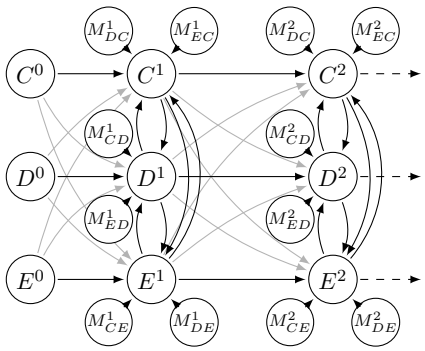


Figure H.2: A causally correctly represented world using an ADBN for Ex. 11. Syntactic DAG constraints of (D)BNs prevented desired cyclic dependencies in this design and "diagonal" state dependencies were enforced (hinted in light grey). In the diagonal case, M^t_{XY} **represents** $M^{t-1\,t}_{XY}$**, i.e.** M^t_{XY} **affects the dependency of state** Y^t **on** X^{t-1}**.**

Notation 3 (Vector Operands). *Let* $\mathbf{f}_\Gamma(\vec{X}, Y)$ *be the product of applied operands* f *to every row* $\{1 \leq i \leq \text{rank}(\vec{X})\}\backslash\Gamma$*, i.e., we iterate over every row of* $\vec{X}$ *without rows in the set* Γ *and apply* f *to this row's elements. Scalars* Y *are used in every row, i.e.,*

$$\mathbf{f}_\Gamma(\vec{X}, Y) = \prod_{i\backslash\Gamma} f(X_i, Y) \qquad \blacktriangle$$

Notation 4 (Lexicographic Order). *Let* $\prec$ *be a lexicographic term order, such that* $X_*^{t-1} \prec X_*^t$, $X_i^t \prec X_{i+1}^t$, *and* $A_{**}^{t-1} \prec A_{**}^t$, $A_{i*}^t \prec A_{(i+1)*}^t$, $A_{ij}^t \prec A_{i(j+1)}^t$, *and* $A_{**}^t \prec X_*^t$, $X_*^{t-1} \prec A_{**}^t$. $\blacktriangle$

Theorem 3 (Bayesian Network Soundness). For every set of combinations of $\vec{\mathcal{A}}^{1:t}$ an ADBN (as in Fig. H.2) corresponds to a Bayesian network, if, for all t, $\vec{\mathcal{A}}^t$ satisfies the new acyclicity constraint:

$$\begin{aligned} \forall x, y, z \in \vec{X}^t : \mathscr{A}(x,z)^t, \mathscr{A}(z,y)^t \rightarrow \mathscr{A}(x,y)^t \\ \neg\exists q : \mathscr{A}(q,q)^t \,, \end{aligned} \tag{H.3}$$

with a function $\mathscr{A}(i,j)^t$ that is defined as

$$\mathscr{A}(i,j)^t = \begin{cases} \textit{false} & \text{if} \quad A_{ij}^t = \neg a_{ij}^t \\ \textit{true} & \text{if} \quad \text{else} \end{cases} .$$

Following the lexicographic order, the joint probability (JP) $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})$ of an ADBN is specified by,

$$\begin{aligned} P(X_1^0) \cdot \ldots \cdot P(X_n^0) \cdot \prod_{i=1}^{t} P(X_1^i | X_2^i, \ldots, X_n^i, A_{21}^i, \ldots, A_{n1}^i, X_1^{i-1}) \cdot \ldots \\ \cdot P(X_n^i | X_1^i, \ldots, X_{n-1}^i, A_{1n}^i, \ldots, A_{(n-1)n}^i, X_n^{i-1}) \cdot P(A_{12}^i) \cdot \ldots \cdot P(A_{n(n-1)}^i) \,, \end{aligned}$$

written for brevity using Not. 3 as

$$\mathbf{P}(\vec{X}^0) \quad \cdot \quad \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \quad \cdot \quad \mathbf{P}(\vec{\mathcal{A}}^i) \quad .$$

As expected, the JP can be defined recursively:

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \quad = \quad P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}) \quad \cdot \quad \mathbf{P}(\vec{X}^t | \vec{X}^{t^\intercal} \backslash \vec{X}^t, A^{\intercal t}, \vec{X}^{t-1}) \quad \cdot \quad \mathbf{P}(\vec{\mathcal{A}}^t) \quad . \quad \blacktriangle \tag{H.4}$$

Informally, Eq. H.3 states that a deactive activator must break open dependency cycles, i.e., the set of possibly active activators forms a directed acyclic graph (DAG).

Proof of Theorem 3. We show that for every set of combinations of $\vec{\mathcal{A}}^{1:t}$ the joint probability stated in Thm. 3 is unique and well-defined, iff for all t the set of $\vec{\mathcal{A}}^t$ obeys Eq. H.3. We show this by reversing conditional independency assumptions in the semantic JP and find the unique topological order of our syntactical graph structure.

We begin with B_0, which can be written as

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X_1^0) \cdot \ldots \cdot P(X_n^0) \cdot \gamma \\ &= P(X_1^0, \ldots, X_n^0) \cdot \gamma = P(\vec{X}^{0^\intercal}) \cdot \gamma \,, \end{aligned} \tag{H.5}$$

with

$$\gamma = \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,.$$

Consecutively, we roll up the joint distribution according to Bayes' chain rule. Considering an extreme case of a set of activators corresponding to Eq. H.3, it is straightforward that under Eq. H.3 there must always $\exists X^1_{E1} : \forall i A^1_{i(E1)} = \neg a^1_{i(E1)}$, such that due to Eq. H.1, the set of activators and previous states uniquely identify the CPT entry and X^1_{E1} becomes independent of all other $\vec{X}^1$, such that the JP can be written as

$$P(\vec{X}^{0\intercal}) \cdot P(X^1_{E1} | *, \vec{A}^{1\intercal}_{E1}, X^0_{E1}) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,. \quad \text{(H.6)}$$

By reversing X^1_{E1}'s conditional independency we can write

$$P(\vec{X}^{0\intercal}) \cdot P(X^1_{E1} | *, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,.$$

Hence, with

$$\begin{aligned} \mathbf{P}(\vec{\mathcal{A}}^t) &= P(A^t_{12}) \cdot \ldots \cdot P(A^t_{1n}) \cdot \ldots \cdot P(A^t_{n1}) \cdot \ldots \cdot P(A^t_{n(n-1)}) \\ &= P(A^t_{12}, \ldots, A^t_{1n}, \ldots, A^t_{n1}, \ldots, A^t_{n(n-1)}) \\ &= P(\vec{\mathcal{A}}^{t\intercal}) \,, \end{aligned}$$

we can combine $P(\vec{X}^{0\intercal})$ with $P(\vec{\mathcal{A}}^{1\intercal})$ to $P(\vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal})$, s.t. the first eliminated state variable X^1_{E1} can be combined to

$$P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,.$$

Consecutively, there $\exists X^1_{E2} : \forall \{i \backslash E1\}\, A^1_{i(E2)} = \neg a^1_{i(E2)}$, s.t.,

$$P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^1_{E2} | *, X^1_{E1}, *, \vec{A}^{1\intercal}_{E2}, X^0_{E2}) \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,,$$

for which we can reverse the conditional independency again and obtain

$$P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^1_{E2} | *, X^1_{E1}, *, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,,$$

which, according to Bayes' chain rule, can be written as

$$P(X^1_{E2}, X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{\intercal 1}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,.$$

Consecutively repeating this process for every further X_{Ei}, where the i^{th} elimination variable is maximally dependent on the previous $(i-1)$ elimination variables, we, henceforth, approach the elimination of X^1_{En}, which is dependent on up to every other $\vec{X}^1$, which are in fact all eliminated variables up to now, s.t.

$$P(X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^1_{En} | X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{A}^{1\intercal}_{En}, X^0_{En}) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \,,$$

for which we can reverse the conditional independency again and combine the JP finally to

$$P(X^1_{En}, X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \quad \cdot \quad \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{\intercal i}, \vec{X}^{i-1}) \quad \cdot \quad \mathbf{P}(\vec{\mathcal{A}}^i) \quad .$$

Indeed, we already obtained a partial topological order $>$ of $X^1_{En} > X^1_{E(n-1)} > \ldots > X^1_{E1} > \vec{\mathcal{A}}^{1\intercal} > \vec{X}^{0\intercal}$. Following this procedure for the remaining t, we finally obtain

$$P(X^t_{E(n-1)}, ..., X^t_{E1}, \vec{\mathcal{A}}^{t\intercal}, ..., X^1_{En}, ..., X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^t_{En} | X^t_{E(n-1)}, \ldots, X^1_{E1}, A^{1\intercal}_{En}, X^{t-1}_{En}) \,.$$

With a final reverse conditional independency assumption,

$$P(X^t_{E(n-1)}, ..., X^t_{E1}, \vec{\mathcal{A}}^{t\intercal}, ..., X^1_{En}, ..., X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^t_{En} | X^t_{E(n-1)}, ..., X^t_{E1}, \vec{\mathcal{A}}^{t\intercal}, ..., X^1_{En}, ..., X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \,,$$

we obtain a complete topological order and a unique JP of

$$P(X^t_{En}, X^t_{E(n-1)}, ..., X^t_{E1}, \vec{\mathcal{A}}^{t\intercal}, ..., X^1_{En}, ..., X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \,. \tag{H.7}$$

We have shown that the claimed JP of Th. 3 in fact is a unique and well-defined JP defining a topological order of a corresponding Bayesian network. □

Informally speaking, this proof shows that in an ADBN an acyclicity constraint can be postponed to an operation phase while assuring soundness with BNs. This means, a BN can actually be a *cyclic* graph, if it is used correctly.

Proposition 3 (Completeness)**.** An ADBN can model any JP. We have shown that in an ADBN all random variables of t can be dependent on each other, as long as during operations only certain combinations of activators are used. This means, that any form of in-time-slice structure can be modeled through adequate specifications of $\vec{\mathcal{A}}^t$. Straightforwardly, this can, if causally needed, be extended to further "diagonal" dependencies between states of consecutive time slices, including activator random variables for those dependencies. Obviously, this does not create cyclic dependencies and thus satisfies Eq. H.3, i.e., is an ADBN. Such an ADBN would contain all possible dependencies between states and leave the option to (de)activate them, meaning, represents the most general form of an Markov-1 ADBN template including all possible Markov-1 DBN structures. Noteworthy, we can directly embed [Pea09]'s do-calculus here using activators. ▲

H.4 Operations

Based on Thm. 3 marginalization is well-defined and filtering, smoothing and prediction (according to the meaning of [Mur02]) can be derived from the joint distribution. We derive those operations while carefully handling novel in-time-slice dependencies and activator random variables.

Notation 5 (Notation for Observations). *Let $\vec{Z}^t \subseteq \vec{X}^t$ be a set of observed and $\vec{\zeta}^t = \vec{X}^t \backslash \vec{Z}^t$ be the corresponding set of not-observed state variables. Let $B^t \subseteq A^t$ be a set of observed activators and $\vec{\mathcal{B}}^t \subseteq \vec{\mathcal{A}}^t$ be the corresponding column vector representation. Likewise, let $\vec{\beta}^t = \vec{\mathcal{A}}^t \backslash \vec{\mathcal{B}}^t$ be the column vector of all not-observed activators. We write $\vec{z}^t$ for $\vec{Z}^t = \vec{z}$ and $\vec{b}^t$ for $\vec{\mathcal{B}}^t = \vec{b}$.*

Every query contradicting with observations, i.e., $\vec{x}^t$ contradicts $\vec{z}^t$, is defined to be of probability 0. Further, every observation in $\vec{z}^t$ uniquely defines its corresponding random variable in $\vec{X}^t$. ▲

H.4.1 Filtering

We calculate the complete conditional joint probability $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ at every timestep, from which every desired filtering operation can be marginalized out. With a normalization factor α, filtering is generally defined from the JP as

$$P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{0:t-1\intercal}} \sum_{\vec{\beta}^{1:t-1\intercal}} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) \,,$$

where all values of variables $\vec{X}^t, \vec{\mathcal{A}}^t$ are defined by the query and variables $\vec{X}^{0:t-1}, \vec{\mathcal{A}}^{1:t-1}$ are defined by either observations in the sets $\vec{z}^{0:t-1}, \vec{b}^{1:t-1}$ or through summation over unobserved variables in $\vec{\zeta}^{0:t-1}, \vec{\beta}^{1:t-1}$.

Definition 33 (Filtering). *Using the recursive definition of the joint probability in Eq. H.4, filtering is efficiently defined as*

$$P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{t-1\intercal}} \sum_{\vec{\beta}^{t-1\intercal}} P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{b}^{1:t-1\intercal}) \cdot \mathbf{P}(\vec{X}^t | \vec{X}^{t\intercal} \backslash \vec{X}^t, A^{\intercal t}, \vec{X}^{t-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^t) \,. \quad ▲ \quad \text{(H.8)}$$

ADBN filtering from $t-1$ to t has time and space complexity $\mathcal{O}(1)$. Every incremental ADBN filtering is n-times faster than performing it in a serialized fashion having further $\mathcal{O}(t)$ space complexity for storing all orders. Further effort would be needed for generating such a serialized order.

Example 12 (Filtering). *With Thm. 3 we can actually model cyclic dependencies as desired in Ex. 11 and build an ADBN for our example as shown in Fig. H.2.*

Say, Don and Earl did pass an initial checkup, but Claire did not. At $t = 1$, we observe a document transfer from Claire to Don, we are unsure about one from Don to Earl, but can neglect all other transfers. As Claire is credulous, we expect her to influence Earl slightly through *Don, expressible in the filtering operation $P(E^1 | \vec{z}^{0:1\intercal}, \vec{b}^{1\intercal})$, with $\vec{z}^{0:1} = (c^0, \neg d^0, \neg e^0)^\intercal$, and $\vec{b}^1 = (m^1_{CD}, \neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC})^\intercal$.*

A diagonal DBN cannot anticipate the indirect influence, because t_1-Earl is influenced by a t_0-Don that has not received a document from Claire. This means our belief in Earl remains at

0 *due to our initial observation of* $\neg e^0$*, i.e.* $P'(E^1|\vec{z}^{0:1\intercal}, \vec{b}^{1\intercal}) = \langle 0, 1\rangle$*. As* $\vec{b}^1$ *fulfills Eq. H.3 we correctly obtain* $P(E^1|\vec{z}^{0:1\intercal}, \vec{b}^{1\intercal}) = \langle 0.32, 0.68\rangle$ *using an ADBN, i.e. we anticipate that Earl is influenced by Claire through Don.*

To achieve the same result in a diagonal DBN, we need observations at a finer time scale, where all indirect influences are serialized, e.g., we must first observe m^1_{CD}*, anticipated in the filtering operation* $P'(E^1|\vec{z}^{0:1\intercal}, \vec{b}^{1\intercal})$ *and then insert a "correcting" "time"-slice* $t = 1.1$*, where we anticipate the possibilities of* M^1_{DE} *in another operation* $P'(E^{1.1}|\vec{z}^{0:1.1\intercal}, \vec{b}^{1:1.1\intercal})$*. To achieve the result of one ADBN operation, we need* $n - 1$ *"diagonal"-operations.* ♦

Prediction is a filtering operation with an empty observation set. However, as a minimal set of observations is needed to remove cycles, plain prediction is not possible in our syntax. However, by splitting a prediction-observation-set into two subsets, e.g. first the lower triangle and second the upper triangle of A^{t+1} is observed to be deactive, prediction becomes possible. While this does not cover all possible chain reactions, it then covers significantly more than a classic DBN could cover (none), as previously discussed in Sec. H.3 and Prop. 2.

H.4.2 Smoothing

Similar to the filtering operation, the general smoothing operation $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}|\vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$, $k < t$ can be derived from the joint probability as

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}|\vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \quad = \quad \alpha \cdot \sum_{\vec{\zeta}^{0:k-1\intercal}} \sum_{\vec{\beta}^{1:k-1\intercal}} \sum_{\vec{\zeta}^{k+1:t\intercal}} \sum_{\vec{\beta}^{k+1:t\intercal}} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) \quad =$$

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}|\vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal}) \cdot P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal}|\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}) \,,$$

in which we find a previous (stored) filtering operation, known as a forward message, and a new latter term commonly known in smoothing operations. Using an adequate recursive definition for the latter term, we obtain an efficient calculation method using a "backward message." The commonly known "sensor model" is, due to in-time-slice dependencies, included in the forward, as well as backward message.

Definition 34 (Smoothing)**.** *Smoothing at timestep* k *considering all evidences up to* t *is defined as*

$$\begin{aligned} P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}|\vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) &= \alpha \cdot P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}|\vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal}) \\ &\cdot \sum_{\vec{\zeta}^{k+1\intercal}} \sum_{\vec{\beta}^{k+1\intercal}} \mathbf{P}(\vec{X}^{k+1}|\vec{X}^{k+1\intercal} \backslash \vec{X}^{k+1}, A^{\intercal k+1}, \vec{X}^k) \cdot \mathbf{P}(\vec{\mathcal{A}}^{k+1}) \\ &\cdot P(\vec{z}^{k+2:t\intercal}, \vec{b}^{k+2:t\intercal}|\vec{X}^{k+1\intercal}, \vec{\mathcal{A}}^{k+1\intercal}) \,. \end{aligned} \tag{H.9}$$

The last term corresponds to the backward message and was calculated in the previous (i.e., previously calculated, but temporally consecutive) smoothing operation. ▲

Performing smoothing over all $k < t$ has $\mathcal{O}(t^2)$ time and constant space complexity or, by storing filtering operations, $\mathcal{O}(t)$ time and space complexity. Compared to a serialized version, without actually serializing, n^2-times faster or n-times faster and smaller.

Example 13 (Explaining away)**.** *Continuing Ex. 12 this example demonstrates that smoothing handles explaining away over multiple timesteps and respects indirect causes. Say, only Don underwent a successful compliance check at time $t = 0$, i.e., $\vec{z}^0 = (\neg d^0)$. For $t = 1$ we found the same document transfer as previously, and for $t = 2$, a Sunday, we can neglect all, i.e., $\vec{\beta}^2 = \emptyset$. On that Sunday also irregularities in Earl's documents were found, i.e., $\vec{z}^2 = (e^2)$.*

If we perform the smoothing operation for Claire's initial belief state without considering evidence from $t = 2$, we end up with our prior belief of $P(C^0|\vec{z}^{0:1\intercal}, \vec{b}^{1\intercal}) = \langle 0.5, 0.5\rangle$, as we have gained no new information. However, with observations from $t = 2$, we need to consider an indirect influence by Claire onto Earl and our belief in her rises to $P(C^0|\vec{z}^{0:2\intercal}, \vec{b}^{1:2\intercal}) \approx \langle 0.532, 0.468\rangle$.

The slow increase is due to our high prior belief in Earl manipulating documents of $P(e^0) = 0.7$ and it is more likely that Earl has been manipulating documents ever since. If, say, Earl can be relieved from initial incriminations, i.e., $\neg e^0$, the only explanation for this situation is an indirect cause of Claire being credulous, which is correctly handled as $P(c^1, m^1_{DE}|\vec{z}^{0:2\intercal}, \vec{b}^{1:2\intercal}) = 1$. We can update our initial prior beliefs using smoothing and find that $P(d^0) = P(e^0) = 0$ but $P(c^0) = 1$. We can now say for sure, Claire is corrupt. ♦

In a classic diagonal (A)DBN the last example would have been unexplainable, as indirect influences of t_1 (causally) would first be anticipated a step later at t_2 (for n=3). The detailed explanation is confusing, because it is not causal: at t_2, the time of incriminating evidence for Earl, we know that Earl is only influenced by himself, i.e. only t_1-Earl can be the source of his credulousness. At t_1, Earl only receives a document from integrous t_0-Don (observation). This is where the problem lies, t_0-Claire should have influenced t_0-Don by now, but t_0-Claire influences t_1-Don with her message m^1_{CD}. I.e. Earl cannot become credulous and the observation e^2 remains unexplainable. Mathematically we obtain $P(e^0|\vec{z}^{0:2\intercal}, \vec{b}^{1:2\intercal}) = 0$ because all terms in this calculation involve either the CPT entry $P(e^2|\underline{\neg m^2_{*E}}, C^1, D^1, \underline{\neg e^1}) = 0$ or $P(e^1|M^1_{DE}, \underline{\neg m^1_{CE}}, C^0, \underline{\neg d^0}, \underline{\neg e^0}) = 0$ (Underlined CPT attributes uniquely identify these entries to be 0). By definition, we obtain $P(\neg e^2|\ldots, e^2, \ldots) = 0$, and, thus, we stand in conflict with the probability axioms of Kolmogorov.

H.5 Discussion

Using an ADBN has the benefit of anticipating indirect causes in-time in an over-the-time evolving process. Still, it comes with a cost of introduced activators, which need to be defined and enforce minimal observation sets of activators (Eq. H.3: any acyclic constellation of probably active activators is allowed). However, we came from a point of view where activators existed and Prop. 2 shows that classic DBNs are significantly more restricted (the number of uniformly directed bipartite graphs with n vertices is far smaller than the number of possible DAGs). The need to anticipate indirect influences originated from coarse observation timesteps, where indirect influences must be anticipated to explain made observations. Where we motivated coarse timesteps from an unavailability of finer observations, the choice of coarser timesteps is also motivated by computational feasibility. Not being bound to the finest available observation granularity relaxes the rate of needed time-updates and is also discussed by [PT05] in the form

of Asynchronous DBNs using [NSK02]'s CTBNs. Still, Asynchronous DBNs and CTBNs run into the same problem given in Prop. 2 of anticipating indirect influences during one timestep.

We have discussed the complexity of operations over *time* and have shown that in an ADBN we obtain the same, and even simpler, complexities as in classic DBNs. However, like in any other DBN, the dimension complexity in terms of nodes of one operation remains computationally intractable and demands approximate inference techniques, which can greatly benefit from context specific independencies, as shown by [BFGK96].

Notwithstanding, it is possible that Eq. H.3 does not hold in a particular situation. Still, we now have a direct indicator for potentially spurious results. In this particular situation a mitigation is needed, but is beyond the scope of this paper. Such a mitigation would be, for example, to move small subsets of activators to a neighboring timestep or enforcing observations of more deactive activators.

H.6 Conclusion

We have shown that indirect causes in dynamic Bayesian networks cause conflicts in representing causality. These conflicts arose from using a modeled dimension for assuring syntactic requirements. By extending dynamic Bayesian networks with activator variables to ADBNs, we are able to move acyclicity constraints from a design phase to a later operation phase. Without the need of algorithm frameworks, degrading a Bayesian network to a reasoning process, we obtained a solid mathematical basis sound to Bayesian networks with a causally correct anticipation of indirect causes in dynamic Bayesian networks under much softer restrictions.

Future work is dedicated to further acyclicity constraints when considering properties of local CPDs, which even allow operations with cyclic activator sets and extensions to relational Bayesian networks [Jae97].

Bibliography

[BFGK96] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context-Specific Independence in Bayesian Networks. In *UAI 1996: 12th Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, pages 115–123, 1996.

[DGH+14] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *KDD 2014: 20th ACM International Conference on Knowledge Discovery and Data Mining, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014.

[FBCC+10] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79, 2010.

[GK95] Sabine Glesner and Daphne Koller. Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases. In *ECSQARU 1995: Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, Fribourg, Switzerland, July 3-5, 1995*, pages 217–226, 1995.

[HHNK95] Peter Haddawy, James Helwig, Liem Ngo, and Robert Krieger. Clinical Simulation using Context-Sensitive Temporal Probability Models. In *Symposium on Computer Applications in Medical Care*, volume 1, pages 203–207, 1995.

[Jae97] Manfred Jaeger. Relational Bayesian Networks. In *UAI 1997: 13th Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pages 266–273, 1997.

[Jae01] Manfred Jaeger. Complex Probabilistic Modeling with Recursive Relational Bayesian Networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):179–220, 2001.

[MM15a] Alexander Motzek and Ralf Möller. Exploiting Innocuousness in Bayesian Networks. In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, ACT, Australia, November 30 - December 4, 2015*, pages 411–423, 2015.

[MM15b] Alexander Motzek and Ralf Möller. Indirect Causes in Dynamic Bayesian Networks Revisited. In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, pages 703–709, 2015.

[Mur02] Kevin Patrick Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[NH97] Liem Ngo and Peter Haddawy. Answering Queries from Context-Sensitive Probabilistic Knowledge Bases. *Theoretical Computer Science*, 171(1-2):147–177, 1997.

[NHH95] Liem Ngo, Peter Haddawy, and James Helwig. A Theoretical Framework for Context-Sensitive Temporal Probability Model Construction with Application to Plan Projection. In *UAI 1995: 11th Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 419–426, 1995.

[NSK02] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Continuous Time Bayesian Networks. In *UAI 2002: 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, pages 378–387, 2002.

[Pea09] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[PR03] Judea Pearl and Stuart Russell. Bayesian Networks. In Michael A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 157–160. MIT Press, 2003.

[PT05] Avi Pfeffer and Terry Tai. Asynchronous Dynamic Bayesian Networks. In *UAI 2005: 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 467–476, 2005.

[SDW05] Sumit Sanghai, Pedro Domingos, and Daniel Weld. Relational Dynamic Bayesian Networks. *Journal of Artificial Intelligence Research*, 24:759–797, 2005.