ON COLLABORATIVE INTRUSION DETECTION

EMMANOUIL VASILOMANOLAKIS

Dissertation
Zur Erlangung des akademischen Grades
Doktor rerum naturalium (Dr. rer. nat.)

genehmigte Dissertationsschrift in englischer Sprache
von MSc. Emmanouil Vasilomanolakis
aus Darmstadt
geboren in Cholargos, Griechenland

Erstreferent: Prof. Dr. Max Mühlhäuser
Korreferent: Prof. Dr. Simin Nadjm-Tehrani
Tag der Einreichung: 10. May 2016
Tag der Prüfung: 11. July 2016

Fachgebiet Telekooperation
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulkennziffer D-17
Darmstadt, 2016

Dedicated to the loving memory of my father
Dr. Michael Vasilomanolakis.

ABSTRACT

CYBER-attacks have nowadays become more frightening than ever before. The growing dependency of our society on networked systems aggravates these threats; from interconnected corporate networks and Industrial Control Systems (ICSs) to smart households, the attack surface for the adversaries is increasing. At the same time, it is becoming evident that the utilization of classic fields of security research alone, e.g., cryptography, or the usage of isolated traditional defense mechanisms, e.g., firewalls and Intrusion Detection Systems (IDSs), is not enough to cope with the imminent security challenges.

To move beyond monolithic approaches and concepts that follow a "cat and mouse" paradigm between the defender and the attacker, cyber-security research requires novel schemes. One such promising approach is *collaborative intrusion detection*. Driven by the lessons learned from cyber-security research over the years, the aforesaid notion attempts to connect two instinctive questions: "if we acknowledge the fact that no security mechanism can detect all attacks, can we beneficially combine multiple approaches to operate together?" and "as the adversaries increasingly collaborate (e.g., Distributed Denial of Service (DDoS) attacks from whichever larger botnets) to achieve their goals, can the defenders beneficially collude too?". Collaborative intrusion detection attempts to address the emerging security challenges by providing methods for IDSs and other security mechanisms (e.g., firewalls and honeypots) to combine their knowledge towards generating a more holistic view of the monitored network.

This thesis improves the state of the art in collaborative intrusion detection in several areas. In particular, the dissertation proposes methods for the detection of complex attacks and the generation of the corresponding intrusion detection signatures. Moreover, a novel approach for the generation of alert datasets is given, which can assist researchers in evaluating intrusion detection algorithms and systems. Furthermore, a method for the construction of communities of collaborative monitoring sensors is given, along with a domain-awareness approach that incorporates an efficient data correlation mechanism. With regard to attacks and countermeasures, a detailed methodology is presented that is focusing on sensor-disclosure attacks in the context of collaborative intrusion detection.

CONTRIBUTIONS The scientific contributions can be structured into the following categories:

ALERT DATA GENERATION This thesis deals with the topic of alert data generation in a twofold manner: first it presents novel approaches for detecting complex attacks towards generating alert signatures for IDSs; second a method for the synthetic generation of alert data is proposed. In particular, a novel security mechanism for mobile devices is proposed that is able to support users in assessing the security status of their networks. The system can detect sophisticated attacks and generate signatures to be utilized by IDSs. The dissertation also touches the topic of synthetic, yet realistic, dataset generation for the evaluation of intrusion detection algorithms and systems; it proposes a novel dynamic dataset generation concept that overcomes the shortcomings of the related work.

COLLABORATIVE INTRUSION DETECTION As a first step, the thesis proposes a novel taxonomy for collaborative intrusion detection accompanied with building blocks for Collaborative IDSs (CIDSs). Moreover, the dissertation deals with the topics of (alert) data correlation and aggregation in the context of CIDSs. For this, a number of novel methods are proposed that aim at improving the clustering of monitoring sensors that exhibit similar traffic patterns. Furthermore, a novel alert correlation approach is presented that can minimize the messaging overhead of a CIDS.

ATTACKS ON CIDSS It is common for research on cyber-defense to switch its perspective, taking on the viewpoint of attackers, trying to anticipate their remedies against novel defense approaches. The thesis follows such an approach by focusing on a certain class of attacks on CIDSs that aim at identifying the network location of the monitoring sensors. In particular, the state of the art is advanced by proposing a novel scheme for the improvement of such attacks. Furthermore, the dissertation proposes novel mitigation techniques to overcome both the state of art and the proposed improved attacks.

EVALUATION All the proposals and methods introduced in the dissertation were evaluated qualitatively, quantitatively and empirically. A comprehensive study of the state of the art in collaborative intrusion detection was conducted via a qualitative approach, identifying research gaps and surveying the related work. To study the effectiveness of the proposed algorithms and systems extensive simulations were utilized. Moreover, the applicability and usability of some of the contributions in the area of alert data generation was additionally supported via Proof of Concepts (PoCs) and prototypes.

The majority of the contributions were published in peer-reviewed journal articles, in book chapters, and in the proceedings of international conferences and workshops.

ZUSAMMENFASSUNG

Cyberangriffe entwickeln sich zu immer ausgeklügelteren Prozessen mit zunehmend schwerwiegenderen Folgen für die Angegriffenen. Gleichzeitig sind immer mehr Aspekte unseres Lebens durch Cyber-Systeme verbunden, werden über diese gesteuert und von diesen beeinflusst – von großen Industrieanalagen über Firmennetzwerke bis hin zu privaten Häusern und Endgeräten. Diese beiden Veränderungen beeinflussen sich gegenseitig und stellen die Forschung an Schutzmaßnahmen vor stets neue Herausforderungen. Klassische Forschungsfelder der Cybersicherheit, wie z.B. die Kryptografie, sowie monolithische und isoliert betriebene Schutzsysteme, wie z.B. Firewalls und Eindringlingserkennungssysteme (intrusion detection systems, IDS), sind in der heutigen Form nicht mehr ausreichend, um mit den neuen Herausforderungen angemessen umzugehen.

Die Arbeit im Feld der Cybersicherheit benötigt neue Ansätze, um bisherige monolithische Schutzsysteme und das weiter beschleunigende Rennen zwischen Angreifern und Verteidigern zu gewinnen. Der Einsatz von kollaborativen Eindringlingserkennungssystemen (collaborative intrusion detection systems, CIDS) ist ein solcher, vielversprechender Ansatz. Die Methodik hinter CIDS fußt auf zwei Grundannahmen: 1. Einzelne Schutzsysteme können niemals alle Angriffe erkennen, daher werden verschiedene Schutzsysteme miteinander kombiniert. 2. Da Angreifer zunehmend kollaborieren (wie beispielsweise bei DDoS-Angriffen oder der Nutzung von Botnets), müssen dies auch Schutzsysteme tun. CIDS begegnen den o.g. neuen Herausforderungen mit neuen Methoden für IDS und für andere Schutzmechanismen (z.B. Firewalls und Honeypots); dabei ist es das Ziel, das Wissen dieser Systeme zu einer umfassenderen Sicht auf das zu überwachende Netzwerk zusammenzufassen.

Diese Dissertation erweitert den Stand der Wissenschaft in der kollaborativen Angriffserkennung in mehreren Bereichen. Insbesondere werden Methoden zur Erkennung komplexer Angriffe sowie die Erzeugung der dazu passenden Angriffs-Signaturen vorgeschlagen. Weiterhin wird auch ein neuartiger Ansatz zur Erstellung von Warn-Datensätzen vorgestellt, welcher Wissenschaftler bei der Evaluierung von zukünftigen IDS-Algorithmen und Systemen unterstützten kann. In Bezug auf Angriffs- und Schutzmechanismen wird eine detaillierte Methodik zur Angriffserkennung mittels Sensoren im Kontext von kollaborativer Angriffserkennung vorgestellt.

BEITRÄGE Die wissenschaftlichen Beiträge dieser Dissertation lassen sich wie folgt kategorisieren:

AGGREGATION VON WARNUNGEN Die Herausforderung der Aggregation von Warnungen wird hier in zwei Schritten bearbeitet: Erstens werden neuartige Ansätze zur Erkennung von komplexen Angriffen einschließlich der Generierung von Signaturen für IDSs vorgestellt. Zweitens wird eine Methode zur Erzeugung von synthetischen Warnungen vorgestellt. Im Detail wird ein neuer Sicherheitsmechanismus für mobile Geräte vorgeschlagen, welcher Benutzer bei der Beurteilung der Sicherheit von Netzwerken unterstützt. Diese Methode kann ausgefeilte Angriffe erkennen und daraus Signaturen erzeugen, welche dann von IDSs verwendet werden können. Diese Dissertation betrachtet auch die Erzeugung von synthetischen, aber dennoch realistischen Datensätzen, welche die Evaluierung von IDS-Algorithmen und Systeme unterstützen. Dazu wird ein neues Konzept zur dynamischen Generierung von Datensätzen verwendet, welches die Einschränkungen der verwandten Arbeiten löst.

KOLLABORATIVE ANGRIFFSERKENNUNG In einem ersten Schritt wird eine neuartige Taxonomie zusammen mit Bausteinen für kollaborative IDSs (CIDSs) vorgestellt. Weiterhin behandelt diese Dissertation die Themen der Korrelation von Warnmeldungen bzw. Daten, sowie deren Aggregation im Kontext von CIDSs. Dafür wird eine Reihe von neuen Methoden vorgeschlagen, die auf eine Verbesserung der Gruppierung (engl. clustering) von Sensoren abzielen, welche ähnliche Muster im Netzwerkverkehr aufweisen. Weiterhin wird ein neuer Ansatz zur Korrelation von Warnungen vorgestellt, welcher den Nachrichten-Overhead von CIDSs verringert.

ANGRIFFE AUF CIDSS Für die Forschung in der Cyber-Sicherheit ist es typisch, einen Perspektivwechsel vorzunehmen, um aus der Sicht der Angreifer die Reaktion auf neue Schutz-Mechanismen zu bestimmen. Diese Dissertation folgt diesem Ansatz und fokussiert dabei auf Klassen von Angriffen gegen CIDSs, die darauf abzielen, die Position von Sensoren im Netzwerk zu bestimmen. Hier wird der Stand der Wissenschaft durch eine Verbesserung dieser Positionsbestimmung erweitert. Weiterhin werden neue Schutzmechanismen vorgestellt, um Angriffe nach dem Stand der Wissenschaft als auch deren Verbesserungen zu unterbinden.

EVALUIERUNG Alle vorgestellten Methoden wurden sowohl qualitativ als auch quantitativ und empirisch evaluiert. Mit dem qualitativen Ansatz wurde eine umfassende Studie über den Stand der Wissenschaft in kollaborativen Angriffserkennungs-Systemen angefertigt, um Forschungsfragen zu identifizieren und verwandte Arbeiten zu

erfassen. Die Effektivität der vorgeschlagenen Algorithmen und Systeme wurde anhand von umfassenden Simulationsstudien gezeigt. Weiterhin wurden die Beiträge im Bereich der Korrelation von Warnungen auf ihre Anwendbarkeit und Benutzbarkeit anhand von Proof of Concepts (PoCs) sowie Prototypen überprüft.

Die meisten hier vorgestellten Beiträge wurden im Peer-Review-Verfahren von Wissenschaftlern geprüft und in Journalen, Buchkapiteln und Tagungsbänden internationaler Konferenzen und Workshops veröffentlicht.

ACKNOWLEDGMENTS

First, I would like to express me warmest thanks to Prof. Dr. Max Mühlhäuser. Max trusted me, gave me the freedom to discover my research interests and always provided excellent and to-the-point advice whenever I felt lost.

Likewise, I would like to thank Prof. Dr. Simin Nadjm-Tehrani for her interest in my research work, her helpful comments and for agreeing in (co-)supervising my dissertation.

Similarly, my work and this thesis would not have come to place without the assistance of Mathias Fischer.

In addition, this thesis would not be the same without the support and encouragement of a great number of TK (and ex-TK) members. I find it hard to imagine a better working environment: Jörg Daubert, Shankar Karuppayah, Carlos Garcia Cordero, Tim Grube, Siavash Valipour, Florian Volk, Sheikh Mahbub Habib, Sascha Hauke, Fábio Borges, and Stefan Schiffner. In the same context, my gratitude to several TK people whose work supported me over the years: Elke Halla, Nina Jäger, Silke Romero, Denny Fuchs, Karin Tillack, Fabian Herrlich, Elke Reimund, and all other wonderful TK members.

Moreover, a big thanks goes to *AGT International* for their full support during my research. In fact, without this support, I would not be able to neither start or finish my studies. Many thanks to Joachim Schaper, Panayotis Kikiras, Alexander Wiesmaier, and all the people I had the honor to work with in AGT.

Furthermore, a very special thanks goes to Katerina Stourna, Ioannis Bouziannis, Panos Protopapas, Michalis Kiparisis (and many others!) for their support and understanding during the last three years of my life. Last, and definitely not least, I want to thank my family for their unconditional love and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ASL Attack Specification Language

CDDHT Cyber Disease DHT

DHT Distributed Hash Table

DIDMA Distributed Intrusion Detection system using Mobile Agents

DOMINO Distributed Overlay for Monitoring Internet Outbreaks

DoS Denial of Service

DDoS Distributed Denial of Service

IDS Intrusion Detection System

INDRA Intrusion Detection and Rapid Action

LarSID Large Scale Intrusion Detection

MA Mobile Agent

MAD Mobile Agents Dispatcher

PKI Public Key Infrastructure

SA Static Agent

SPoF Single Point of Failure

TTP Trusted Third Party

VPN Virtual Private Network

P2P Peer-to-Peer

CRIM Cooperative Intrusion Detection Framework

CIDS Collaborative IDS

AIDS Anomaly IDS

ANIDS Anomaly Network IDS

PII Personally Identifiable Information

DIDS Distributed Intrusion Detection System

APT Advanced Persistent Threat

IDMEF Intrusion Detection Message Exchange Format

DNS Domain Name System

URI Uniform Resource Identifier

CLB Constrained Load Balancing

LGPL Lesser General Public License

PRA Probe Response Attack

PREPARE Probe REsPonse Attack fRamEwork

UI User Interface

GMEM Generic Marker Encoding Methodology

GrIDS Graph Based Intrusion Detection System

AAFID Autonomous Agents For Intrusion Detection

EMERALD Event Monitoring Enabling Responses to Anomalous Disturbances

HIDE Hierarchical Intrusion Detection

IDA Intrusion Detection Agent

AA Alerting Agents

PHAD Packet Header Anomaly Detector

LERAD Learning Rules for Anomaly Detection

FA False Alarm

VM Virtual Machine

CA Certificate Authority

EFSM Extended Finite State Machine

FSM Finite State Machine

R/W Read/Write

ICS Industrial Control System

PLC Programmable Logic Controller

RTU Remote Terminal Unit

SPLD Single-Protocol Level Detection

PLD Payload Level Detection

MSLD Multi-Stage Level Detection

IoT Internet of Things

ID2T Intrusion Detection Dataset Toolkit

FLAME Flow-Level Anomaly Modeling Engine

PoD Ping of Death

OS Operating System

TTL Time To Live

GUI Graphical User Interface

VOIP Voice Over IP

IPS Intrusion Prevention System

PCI Protocol Control Information

PoC Proof of Concept

SNMP Simple Network Management Protocol

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

SIP Session Initiation Protocol

Part I

PREFACE

The first part of the thesis aims at introducing the reader to the topic and supporting her by discussing the required background knowledge and definitions that will be used in the course of the thesis. Chapter 1 serves as an introduction to the motivation of the thesis as well as to its structure. Chapter 2 provides the reader with background knowledge of various topics that are fundamental for the understanding of the remainder of the thesis. Chapter 3 touches the topic of collaborative intrusion detection and respective terms that will be utilized in the thesis. Lastly, Chapter 4 proposes a novel taxonomy for collaborative intrusion detection and conducts a detailed survey of the related work.

1 INTRODUCTION

The only truly secure system is one
that is powered off, cast in a block of concrete
and sealed in a lead-lined room with armed guards.

— Gene Spafford

RESEARCH in the area of cyber-security has increased significantly over the last years. This fact however is neither random nor unforeseen; rather it is highly connected to the massive increase in the number and sophistication of cyber-attacks. Researchers nowadays have realized that existing and future infrastructures, designed by humans, cannot be perfectly secure. In this context, research towards novel cyber-security systems is fundamental for the protection and the resilience of networks. In addition, regardless of the specifics of a newly proposed security mechanism, researchers are obliged to evaluate their ideas in a scientific, transparent and broadly acceptable manner. However, as it will become evident in the course of this thesis, this evaluation procedure is not at all trivial; in fact it raises several additional research challenges.

This Ph.D. thesis contributes in the field of cyber-security. In particular, the dissertation significantly improves the state of the art in the areas of alert data generation and intrusion detection with a twofold vision. First, it *improves the techniques for generating alert data and hence evaluating security mechanisms* and second it *significantly advances the field of collaborative intrusion detection*.

This introductory chapter is structured as follows. Section 1.1 describes the motivation, the setting and the research questions that this thesis is addressing. Afterwards, Section 1.2 gives the reader a summary of the contributions, while Section 1.3 tabulates the corresponding publications that have been made as a result of the conducted research. Lastly, the structure of the thesis is given in Section 1.4.

PROBLEM STATEMENT

The dependency of our society on networked computers has become frightening: In the economy, all-digital networks have turned from facilitators to drivers; as cyber-physical systems are coming of age, computer networks are now becoming the central nervous systems of our physical world – even of highly critical infrastructures such as the power grid. At the same time, the 24/7 availability and correct functioning of networked computers has become much more threatened: The number of sophisticated and highly tailored attacks on IT systems has significantly increased [148]. The monetary damage of such attacks is also substantial (see Figure 1).

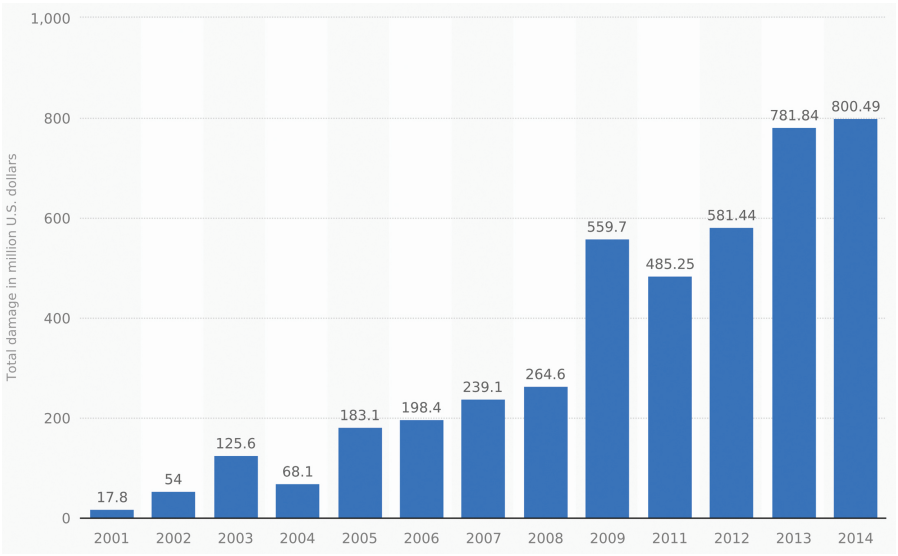


Figure 1: Amount of monetary damage caused by reported cyber crime from 2001 to 2014 (in million U.S. dollars). Data taken from the Internet Crime Complain Center (IC3) and Statista.

Furthermore, emerging technology trends such as the concept of the IoT [61] and/or the ongoing convergence of ICSs with the Internet, further highlight the challenges that need to be tackled from a security perspective. In fact, the author's preliminary work towards a security analysis of existing or proposed IoT architectures suggests that the topics of security and privacy are yet to be achieved [60, 170]. As the size of the IoT market (see Figure 2) is planned to grow exponentially, it is to be expected that the total number of cyber attacks will further increase.

Intrusion Detection Systems (IDSs) are a key component of the corresponding defense measures nowadays; they have been extensively studied and utilized in the past [87, 49]. However, since conventional IDSs are not scalable to big company networks and beyond, nor to massively parallel attacks, Collaborative IDSs (CIDSs) have emerged [21, 196]. Such systems consist of several monitoring components that

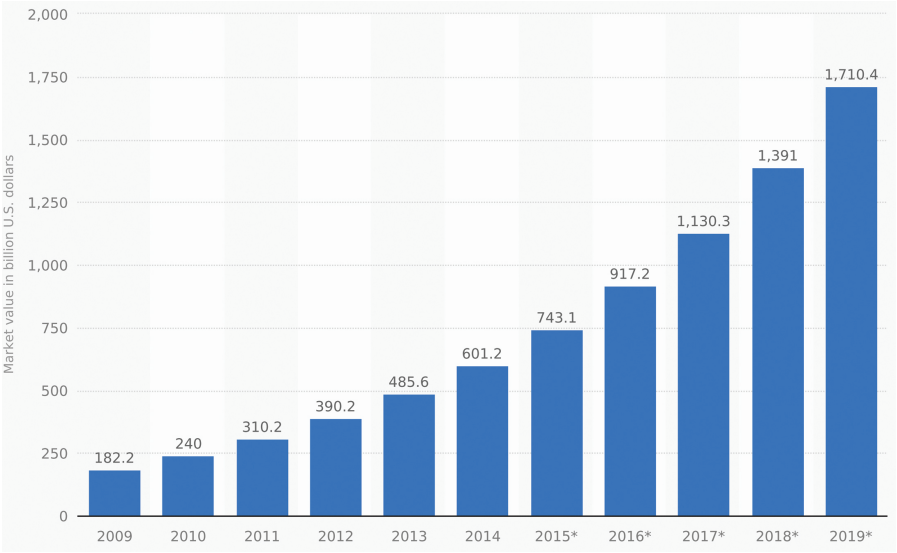


Figure 2: Size of the global IoT market from 2009 to 2019 (in billion U.S. dollars). A star (*) indicates an estimated value. Data taken from HKExnews and Statista.

collect and exchange data. Depending on the specific CIDS architecture, central or distributed analysis components mine the gathered data to identify attacks. Resulting alerts are correlated among multiple monitors in order to create a holistic view of the network monitored.

The novel character of the CIDS area results to many research questions and gaps [196, 172, 107] that will be in-depth discussed in this dissertation. At a glance, the field combines research from many different areas of computer science that include but are not limited to: intrusion detection algorithms (e.g., novel anomaly-based detection algorithms [25]), distributed architectures (e.g., Peer-to-Peer (P2P) systems [96]), data correlation and aggregation (e.g., similarity-based alert correlation algorithms [110]), mechanisms for data dissemination (e.g., gossiping in distributed environments [76]), etc.

With respect to the core areas of CIDSs this thesis is attempting to answer the following research questions:

- How can the collaboration part of a CIDS be practically realized and which parameters influence such a process?
- How effective is the exchange of data on the detection, rather than the alert level, in the context of collaborative anomaly-based detection?
- How can a CIDS be practically realized in large networks that enforce (security policy) restrictions to the flow of data on different sub-networks?

- What kind of novel and efficient correlation mechanisms can be designed for minimizing the communication overhead of a CIDS?
- What is the impact of CIDS disclosure attacks, and how can their design, execution and mitigation be realistically improved?

Apart from the aforementioned research questions, the author's work in the area of CIDSs revealed the need for additional work towards holistically approaching the collaborative intrusion detection field. First, there is a need for novel detection mechanisms, that are able to enhance the state of the art, in the area of ICSs. Furthermore, the problem of creating datasets, that are realistic enough to be utilized for the evaluation of intrusion detection systems and algorithms, has not be tackled so far in the related work. In this sense, the second part of the dissertation is emphasizing in the following corresponding research questions. As the reader may notice, these are highly connected to the topics of alert data generation and evaluation mechanisms for intrusion detection systems and algorithms.

- How can novel and previously unknown attacks that target critical infrastructure, e.g., ICSs, be modeled and identified?
- Can the knowledge gained from the previous research question be utilized to support existing IDSs' infrastructure?
- Is it possible for a security mechanism, such as a honeypot, to detect complex attacks that involve multiple adversarial steps?
- How likely is it for an adversary to evade such a security mechanism?
- Which are the parameters in a synthetically generated dataset, intended for intrusion detection, that can reveal its artificial nature?

THESIS CONTRIBUTIONS

This thesis contributes in several key areas of collaborative intrusion detection and alert data generation. An overview of the structure of the dissertation along with the different contributions is depicted in Figure 3. This section briefly summarizes the various contributions per chapter along with the respective scientific publications. Afterwards, Section 1.3 provides the reader a detailed overview of the publications that are related to the thesis at hand.

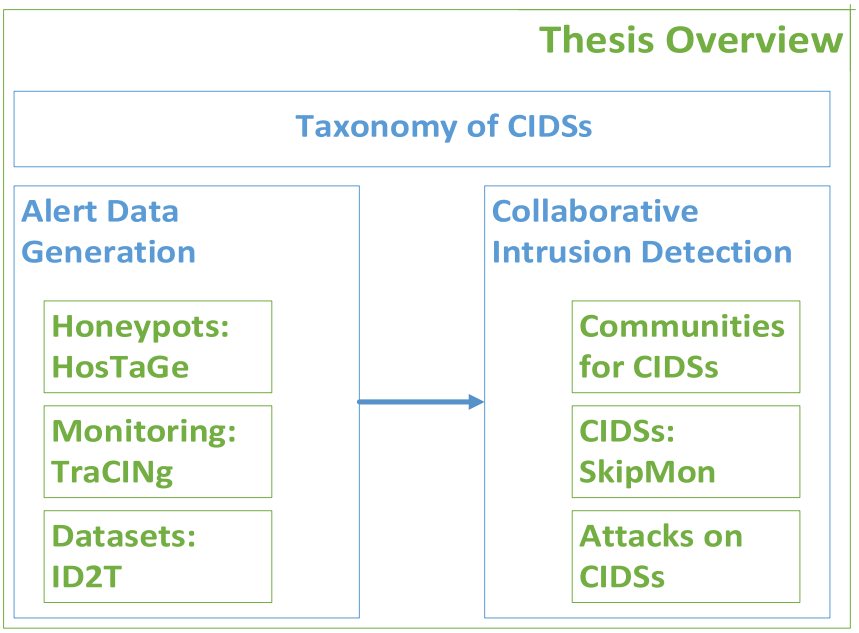


Figure 3: Thesis overview and contributions.

Taxonomy and Survey of CIDSs

The first contribution of this thesis is given in Chapter 4, in which a novel taxonomy for CIDSs is proposed, accompanied by a comprehensive survey and qualitative comparison of the state of the art [172, 167]. To the best of the knowledge of the author, this is the most holistic and up to date study of CIDSs. In addition, this chapter serves as additional motivation for the thesis, in the sense that none of the existing systems can neither fulfill all the requirements proposed in Chapter 3, nor depict an architecture that is suitable for the protection of large network infrastructures. Furthermore, the study of the related work highlights the need for more efficient, realistic and broadly acceptable means for the evaluation of such systems.

Alert Data Generation

The second part of this dissertation copes with the topic of alert data generation. In more details, this subject and the challenge of creating mechanisms that are able to detect attacks, correlate events and produce usable alert data and datasets is firstly touched in Chapter 5. Here, the thesis introduces a security tool for mobile devices, namely *HosTaGe*, which is able to detect malicious behavior, study the adversaries' techniques, correlate attacks, and generate respective signatures for the identified attacks [168, 169, 174, 177]. Furthermore,

the proposed system deals with attacks related to the IoT and ICSs [152][1].

Chapter 6 presents *TraCINg*, a cyber incident monitor that takes as input data generated by monitoring sensors [171]. Among others, *TraCINg* makes use of *HosTaGe* honeypots, hence creating a collaborative platform that is able to aggregate the results of sensors from all over the world. Moreover, the chapter presents a long-term study of deployment of the cyber incident monitor to examine current attack trends and analyze the adversarial behavior. The study also emphasizes in identifying correlated attacks that target more than one of the utilized sensors within a certain time window.

Lastly, Chapter 7 proposes a toolkit for the generation of synthetic, yet realistic, datasets for the evaluation of intrusion detection systems and algorithms [29, 176][108][1]. First, a study of the related work highlights the challenges for the design of such a system and the parameters and properties which can introduce artifacts in a generated dataset. The proposed system, namely the Intrusion Detection Dataset Toolkit (ID2T), exhibits a flexible architecture offering a methodological approach for injecting network files with cyberattacks to generate labeled datasets.

Collaborative Intrusion Detection

The third part of the dissertation copes with the core area of collaborative intrusion detection. In Chapter 8, the thesis presents the idea of *communities* of sensors that collaborate by exchanging features of network traffic to towards creating holistic models of the monitored networks [30]. This way it is possible to generate normality models to be used by anomaly-based detection algorithms. The proposed CIDS concept of this chapter is also the first one that shifts from the alert level exchange to the detection level. To practically realize this concept two stochastic algorithms are developed for grouping monitoring sensors to communities.

Chapter 9 builds on top of the communities idea, presented in the previous chapter. That is, a novel fully distributed CIDS, namely *SkipMon*, is proposed [173] [84][1]. *SkipMon* improves the related work in a multitude of ways. The system is the first one to fulfill the domain awareness requirement, i.e., the ability to dynamically constrain alert dissemination with respect to security policies. Second, a novel similarity-based correlation mechanism is proposed that can effectively correlate large amounts of alert data. The latter, works on the basis of bloom filters which can also be beneficial in terms of the privacy of the alert data. By making use of such a correlation mechanism the system is able to identify and connect sensors that experience similar traffic patterns.

1 This work is a Master thesis supervised by the author of this thesis.

Finally, Chapter 10 introduces contributions in the area of disclosure attacks to CIDSs. Specifically, the chapter deals with a particular class of attacks, called Probe Response Attacks (PRAs), which can be utilized for identifying the network location of monitoring sensors. The thesis proposes an open-source framework that enables the development, improvement, execution and detection of PRAs [175][154][1]. In addition, a novel technique is proposed for the design of such attacks that enables the execution of PRAs with a significantly less time required. Lastly, a multitude of techniques that focus on the detection and mitigation of the attacks are proposed and examined.

PUBLICATIONS

Some parts of this thesis have been published in international peer-reviewed journals, book chapters, conferences and workshops. The following is a detailed list of these publications for each of the main chapters of the thesis.

CHAPTER 1 (General overview and motivation)

- Shankar Karuppayah, *Emmanouil Vasilomanolakis*, Steffen Haas, Max Mühlhäuser, Mathias Fischer: BoobyTrap: On Autonomously Detecting and Characterizing Crawlers in P2P Botnets. CISS@ICC 2016, IEEE [75]
- *Emmanouil Vasilomanolakis*, Jörg Daubert, Manisha Luthra, Vangelis Gazis, Alexander Wiesmaier, Panagiotis Kikiras: On the security and privacy of Internet of Things architectures and systems. SIoT@ESORICS 2015 [170]
- Vangelis Gazis, Manuel Görtz, Marco Huber, Alessandro Leonardi, Kostas Mathioudakis, Alexander Wiesmaier, Florian Zeiger, *Emmanouil Vasilomanolakis*: A survey of technologies for the Internet of things. IWCMC 2015: 1090-1095 [61]
- Vangelis Gazis, Carlos Garcia Cordero, *Emmanouil Vasilomanolakis*, Panayotis Kikiras, Alex Wiesmaier: Security Perspectives for Collaborative Data Acquisition in the Internet of Things. SaSeIoT 2014, Springer LNCS 151: 271-282 [60]

CHAPTER 4

- *Emmanouil Vasilomanolakis*, Shankar Karuppayah, Max Mühlhäuser, Mathias Fischer: Taxonomy and Survey of Collaborative Intrusion Detection. ACM Comput. Surv. 47(4): 55 (2015) [172]
- *Emmanouil Vasilomanolakis*, Mathias Fischer, Max Mühlhäuser, Peter Ebinger, Panayotis Kikiras, Sebastian Schmerl: Collaborative Intrusion Detection in Smart Energy Grids. ICS-CSR 2013: 97-100 [167]

CHAPTER 5

- *Emmanouil Vasilomanolakis*, Shreyas Srinivasa, Carlos Garcia Cordero, Max Mühlhäuser: Multi-stage Attack Detection and Signature Generation with ICS Honeypots. IEEE/IFIP DISSECT 2016 [177]
- *Emmanouil Vasilomanolakis*, Shreyas Srinivasa, Max Mühlhäuser: Did you really hack a nuclear power plant? An industrial control mobile honeypot. IEEE CNS 2015: 729-730 [174]
- *Emmanouil Vasilomanolakis*, Shankar Karuppayah, Max Mühlhäuser, Mathias Fischer: HosTaGe: a Mobile Honeypot for Collaborative Defense. ACM SIN 2014: 330-334 [169]
- *Emmanouil Vasilomanolakis*, Shankar Karuppayah, Mathias Fischer, Max Mühlhäuser, Mihai Plasoianu, Lars Pandikow, Wulf Pfeiffer: This network is infected: HosTaGe - a low-interaction honeypot for mobile devices. SPSM@CCS 2013: 43-48 [168]

CHAPTER 6

- *Emmanouil Vasilomanolakis*, Shankar Karuppayah, Panayotis Kikiras, Max Mühlhäuser: A honeypot-driven cyber incident monitor: lessons learned and steps ahead. ACM SIN 2015: 158-164 [171]

CHAPTER 7

- *Emmanouil Vasilomanolakis*, Carlos Garcia Cordero, Nikolay Milanov, Max Mühlhäuser: Towards the creation of synthetic, yet realistic, intrusion detection datasets. IEEE/IFIP DISSECT 2016 [176] (*Best paper award*)
- Carlos Garcia Cordero, *Emmanouil Vasilomanolakis*, Nikolay Milanov, Christian Koch, David Hausheer, Max Mühlhäuser: ID2T: A DIY dataset creation toolkit for Intrusion Detection Systems. IEEE CNS 2015: 739-740 [29]

CHAPTER 8

- Carlos Garcia Cordero, *Emmanouil Vasilomanolakis*, Max Mühlhäuser, Mathias Fischer: Community-based Collaborative Intrusion Detection. ATCS@SecureComm 2015: 665-681 [30]

CHAPTER 9

- *Emmanouil Vasilomanolakis*, Matthias Kruegl, Carlos Garcia Cordero, Max Mühlhäuser, Mathias Fischer: SkipMon: a Locality-Aware Collaborative Intrusion Detection System. IEEE IPCCC 2015 [173]

CHAPTER 10

- *Emmanouil Vasilomanolakis*, Michael Stahn, Carlos Garcia Cordero, Max Mühlhäuser: Probe-response attacks on collaborative intrusion detection systems: Effectiveness and countermeasures. IEEE CNS 2015: 699-700 [175]

- *Emmanouil Vasilomanolakis*, Michael Stahn, Carlos Garcia Cordero, Max Mühlhäuser: On Probe-Response Attacks in Collaborative Intrusion Detection Systems. IEEE CNS 2016 (to appear) [178]

THESIS OUTLINE

The thesis consists of a total of *eleven chapters* and is logically split into *four parts*, namely the *preface*, the *alert data generation*, the *collaborative intrusion detection* and lastly the *epilogue*. A detailed overview of the structure of the dissertation is given in the following.

- First, Chapter 2 provides the reader with the necessary background in the areas of IDSs, honeypots and on the existing evaluation techniques for IDSs.

- Similarly, Chapter 3 acts as an introduction into the topic of collaborative intrusion detection as well as on the available attacks on CIDSs.

- In Chapter 4, the thesis proposes a novel taxonomy for CIDSs and subsequently discusses the state of the art in a comprehensive manner.

- With Chapter 5, the dissertation enters its second part that is related to alert data generation techniques and mechanisms. In particular, the chapter introduces *HosTaGe* a novel mobile honeypot that emphasizes on the detection of sophisticated attacks in a plethora of scenarios.

- Moreover, Chapter 6 discusses *TraCINg* a cyber incident monitor that makes use of sensors, e.g., *HosTaGe* honeypots, along with a discussion of the results of a long period of deployment.

- In Chapter 7, the thesis discusses in more detail the topic of intrusion detection systems and algorithms evaluation. For this it proposes ID2T a toolkit for the generation of synthetic, yet realist, intrusion detection datasets.

- With Chapter 8, the dissertation enters the core topics of collaborative intrusion detection. In particular the chapter proposes the concept of *communities* of collaborative sensors and respective algorithms for their creation.

- Furthermore, Chapter 9, builds on the basis of the communities concept and introduces *SkipMon* a fully distributed CIDS that offers novel properties in the field of collaborative intrusion detection.
- Chapter 10 deals with the topic of Probe Response Attacks (PRAs) and provides several contributions in their development, detection and mitigation.
- Finally, Chapter 11 concludes this dissertation and provides insights of possible future work in the areas of alert data generation and collaborative intrusion detection.

Readers that are familiar with the topics of IDSs and/or CIDSs can choose to skip Chapters 2 and 3 respectively. Nevertheless, the author encourages the study of Chapter 4 as the various introduced terms and concepts are utilized throughout the whole thesis. The two parts of the dissertation, namely *alert data creation* and *collaborative intrusion detection*, are mostly autonomous and self-contained. In this sense readers that are specifically interested in only one of these can skip directly to the corresponding part. However, interconnections between chapters of different parts do exist; for instance, the introduction of *TraCINg* in Chapter 6 might assist the reader to fully comprehend Chapter 10 and so forth.

2

BACKGROUND

This chapter is intended for providing the reader with an introduction and background knowledge of various topics that will be further discussed in the rest of the thesis. Section 2.1 discusses Intrusion Detection Systems (IDSs), a central topic of this thesis, along with some initial classifications and definitions with regard to them. Section 2.2 deals with honeypots, i.e., an additional detection mechanism that can significantly improve existing intrusion detection mechanisms. Lastly, Section 2.3 deals with the topic of evaluating IDSs with an emphasis on corresponding datasets as well as on tools and mechanisms for creating such datasets. As it will become evident in the course of this thesis, the latter topic of IDS-specific evaluation methods remains a big challenge in the respective research field.

INTRUSION DETECTION SYSTEMS (IDSS)

Besides implementing software and designing hardware to be as secure as possible, it is inevitable that IT systems must be continuously monitored, to ensure their correct functioning, for any kind of anomalies or for signs of intrusions and attacks. Thus, the monitoring process provides an additional line of defense for any kind of (critical) network infrastructure and IT system. Such a task is usually taken over by Intrusion Detection Systems (IDSs) [112, 9].

Classifications and definitions

An IDS monitors a host or a network and analyzes it for signs of intrusions manifesting malicious behavior or security policy violations. Thus, its goal is the detection of any attempt to compromise the confidentiality, integrity, availability of information, or simply to bypass the security mechanisms of a computer or network [12]. The following section presents the most common classifications and definitions for IDSs and briefly summarizes the related work in anomaly detection algorithms.

Deployment position

In regard to the position of their deployment, IDSs can be divided into either *host*-based or *network*-based IDSs [72, 87]. Host-based IDSs analyze events and the behavior of users on the granularity of single devices. This allows collecting detailed information, but introduces additional computational overhead that can affect the overall performance of the monitored system. Moreover, this requires a deployment of IDSs on all devices to be protected. In contrast, network-based IDSs can protect several devices or even entire networks at once, as they monitor only network traffic. More recent work in IDSs also introduced systems that move beyond the aforementioned binary classification. Such work includes proposals towards wireless-based IDSs [20], network behavior analysis, mixed IDSs [89, 156] and even cloud-based approaches [114].

Detection mechanisms

IDSs can be further categorized according to their deployed detection mechanisms into *signature*-based (or misuse-based) and *anomaly*-based[1].

SIGNATURE-BASED DETECTION Signature-based IDSs search for signatures of known attacks and detect their occurrence in the net-

1 Additional classes, e.g., stateful protocol analysis [89], can be considered part of the anomaly class.

work. Such a detection mechanism, however, implies the existence of signatures and therefore is not suitable for identifying novel adversarial behavior. Nevertheless, such IDSs are very commonly utilized for the monitoring of small to medium sized operational networks due to their high precision. Well known examples of such signature-based IDSs include Snort [135] and Suricata [3].

ANOMALY-BASED DETECTION Anomaly-based IDSs attempt to initially learn the normal system state and afterwards define any deviating behavior as an intrusion [9, 25]. In contrast to a signature-based detection, an anomaly-based detection can also detect unknown attacks. However, usually this comes at the cost of a high *false positive* rate, while a signature-based detection usually results in more *false negatives*. The following description of related work with an emphasis on anomaly-based detection in categorical data and rule deduction techniques serves as an introduction towards the work described in Chapter 8.

Discovering anomalies in categorical data is of particular interest to anomaly-based IDSs as they heavily rely on the analysis of categorical attributes [25]. For example, IP addresses are normally represented as categorical rather than numerical attributes. This is an important issue to take into account as not every machine learning technique is able to work well with network data. There are, however, many machine learning algorithms that are well suited for this task.

Mahoney and Chan published the Packet Header Anomaly Detector (PHAD) algorithm [103]. It focuses on finding rules describing the normal appearance of the Ethernet, IP, TCP, UDP, and ICMP protocols. Detection of anomalies in this context is limited to packets not adhering to one of the learned protocols. The algorithm evolved, by taking into account the application layer of the OSI model, into [100], which uses features extracted from TCP streams to model user defined conditional rules. Learning Rules for Anomaly Detection (LERAD) [103], finds rules on its own through a stochastic sampling algorithm. Instead of modeling hand picked rules, LERAD is capable of finding a subset of effective conditional rules that describe normal network data. Due to its latter property, LERAD will be utilized as a detection engine in one of the proposed collaborative intrusion detection concepts in the course of this thesis (see Chapter 8.3.2).

Passive and active monitoring

Traditionally, IDSs are considered as passive monitoring. Even though there are cases, e.g., systems that also act as Intrusion Prevention Systems (IPSs), on which IDS do not have a completely passive operation, overall the detection process is based on passive components. Nevertheless, the process of monitoring of networks can additionally em-

ploy more active components such as honeypots (see following section). This thesis will be utilizing both passive (e.g., in Chapter 8) and active (e.g., in Chapter 5) components for the monitoring process.

HONEYPOTS

Honeypots are systems, whose value lies in being probed, attacked, or compromised [151], and they can provide a more active line of defense compared to passive intrusion detection. As honeypots do not have any production value, any interaction with them, i.e., any incoming communication, is by definition considered an attack [150]. As such, they exhibit a low false positive ratio and their usage can assist in increasing the overall detection accuracy of a IDS [72]. In addition, many honeypots exhibit the ability of automatically capturing the payload of an attack. Hence, they are able to provide additional knowledge regarding recent malware techniques and trends. Furthermore, the utilization of honeypots can assist in reducing the overall attack surface of a network and can additionally serve as the basis for studying the behavior of the adversaries.

This section provides the reader with a brief description of the most common classification with regard to honeypots (i.e., the interaction level). Moreover, a discussion of the state of the art in honeypots is given that serves as a basis for the comprehension of the thesis' proposals in Chapter 5.

INTERACTION LEVEL Honeypots can be classified with respect to the level of interaction that is offered to the attacker, into *low* and *high* interaction[2]. On the one hand, a high-interaction honeypot is essentially a full functional system that exhibits certain vulnerabilities and is closely monitored. Hence, such honeypots need to be carefully safeguarded to avoid a full compromise, which results in an overwhelming effort from the defenders' perspective. On the other hand, a low-interaction honeypot only simulates network operations, usually at the TCP/IP stack. This thesis focuses solely on low-interaction honeypots for a number of reasons. First, they require low resources which makes them suitable for the deployment into constrained devices, e.g., mobile phones. Moreover, this class of honeypots can be efficiently designed such that it is possible to subsequently include emulation support for new protocols. In this context, the reader may refer to Chapter 5, which presents contributions in both the areas of mobile honeypots and ICSs, towards the detection of attacks and the generation of alert data.

2 This thesis considers the so-called *medium* interaction class as a sub-class of the low-interaction since by definition honeypots that claim to be part of this category, e.g., Kippo, still emulate protocols.

ALERT SIGNATURE GENERATION Recently, there have been various proposals in the area of alert signature generation that make use of honeypots (e.g., [80, 162, 78]). For instance, HoneyComb [80] is a system that makes use of the *honeyd* honeypot [127] to generate alert signatures. It has the advantage of only using honeypot network traffic and thus reducing false positives. However, as a result of utilizing honeyd, only high level TCP or UDP information can be examined, making it unsuitable for payload-level analysis.

USABILITY OF HONEYPOTS Most honeypot proposals are strictly focused on implementing novel detection methods. Hence, *user-centric* solutions are not the primary focus and as such the main intended user class is security professionals and not ordinary users. However, the idea and the benefits of creating more user-friendly honeypots is getting more attention lately. In [5, 6] the "Honey@home" is proposed, where organizations and individuals can participate by deploying a honeypot that reports to a large-scale centralized honeypot monitoring system. Nevertheless, this approach does not include mobile devices and there are no clear benefits for the end-user to participate.

With regard to further related work, honeypots have been studied extensively over the recent years [151]. In particular, the low-interaction class exhibits a variety of proposals and implementations, e.g., [10, 127, 140, 123, 63]. In the following, this chapter briefly discusses related work in low-interaction honeypots with a focus on mobile systems. In addition, an introduction to honeypots that emphasize on ICSs protocols is given. This discussion serves as background for the work described in Chapter 5.

Mobile Honeypots

Early work that considered honeypots in the context of mobile devices focused only on Bluetooth communications [56, 198]. However, the recent advances on mobile devices, their interconnectivity as well as their popularity created a whole new ecosystem for honeypot researchers. Existing work in this direction, e.g., [116, 185, 90, 186], usually focuses on the detection of mobile-specific malware. Specifically, Mulliner et al. were the first to discuss the idea of a honeypot for smartphones, by providing initial ideas, challenges and an architecture for their proposed system [116]. Moreover, in [185, 186] Wahlisch et al. provided insights from the deployment of a honeypot on a mobile network. However, their honeypot is not specifically crafted for mobile devices, but rather deployed existing Linux-based desktop honeypots in mobile networks. Furthermore, in [90] the idea of nomadic honeypots has been introduced. The authors focus on mobile-specific attacks and also require their system to collect a large amount of personal information.

Honeypots for ICSs

In their recent work, Minn et al. [109] proposed *IoTPOT*, a honeypot that emphasizes on IoT devices by emulating the Telnet protocol. Their results show an increase of attacks on Telnet that target IoT devices, which, as discussed in Chapter 6, also corresponds to our findings [171]. However, their focus is limited only into Telnet-based attacks (and different CPU architectures). Conpot [134] is another low interaction honeypot that focuses on emulating server side ICSs. Conpot was one of the first honeypots detecting ICS network attacks and is considered the state-of-the-art in this area. Conpot has a few disadvantages however; first, it does not support the emulation of Telnet, and the information that is logged, e.g., for Modbus attacks, is not always sufficient for an in-depth analysis of an attack.

Summary

There are many research challenges and gaps in the area of mobile honeypots and in systems that aim to monitor ICS networks. Furthermore, honeypots are not intended as a stand alone security mechanism, but rather as a complementary approach to improve IDSs. Hence, research in the topic of honeypot-based signature generation (that can be afterwards utilized by IDSs) is required. Chapter 5 will introduce a novel mobile low-interaction honeypot that combines the field of alert signature generation with the ability to monitor ICS networks.

EVALUATING IDSS

The evaluation of intrusion detection algorithms and systems is a topic that exhibits, as it will be shown later in this section, a plethora of challenges for the scientific community. To evaluate their work, researchers can make use of existing datasets, develop their own or utilize specialized tools that are able to generate corresponding datasets. This section discusses the state of the art in this area. In this sense the current section additionally acts as motivation for the second part of this dissertation; the problems and research gaps described in the following suggest the need for novel mechanisms for generating real world alert data (cf. Chapters 5 and 6) as well as for tools that are able to generate realistic datasets for IDSs (cf. Chapter 7).

IDS-Specific Datasets

A number of synthetic and non-synthetic datasets have been published over the years with the purpose of being utilized for the evaluation of intrusion detection algorithms and systems [92, 52, 139, 79].

However, there are two major problems with most existing datasets. First, many of them have been created over a decade ago and thus do not exhibit realistic network traffic nor contain up to date cyber-attacks. Furthermore, as a result of the synthetic creation of the datasets, the majority of these include undesired artifacts that can significantly reduce their usability.

The DARPA 1999 dataset [92] is an illustrative example of both the aforementioned problems. Being generated more than 15 years ago, it contains network traffic and attacks that are antiquated. Moreover, a lot of criticism has been made with regard to undesired artifacts during the generation of the attacks [106]. For instance, Mahoney and Chan [101] discovered inconsistencies in the TTL values of malicious and non-malicious traffic. Due to such inconsistencies the evaluation results of various anomaly-based detection algorithms might be misinterpreted as a result of them identifying the artifacts, rather than the actual attacks, that are present in a dataset. Nevertheless, it should be noted that the dataset is still being utilized for the evaluation of various recent ensemble-based approaches (e.g., [121, 16]).

Other examples of datasets include [1, 31, 161] and [52]. Nevertheless, all of these datasets exhibit problems such as the absence of flow data and ground truth knowledge, availability issues, etc. [79]. For instance, the MAWI dataset [52] consists of a very large number of modern network captures and can be, indeed, particularly useful for various research purposes but it does not contain any ground truth nor packet payloads.

Dynamic Creation of Datasets

Beyond single, static datasets, research has been conducted in the area of generating datasets dynamically, e.g., [144, 17]. For instance, Shiravi et al. [144] proposed a systematic approach for generating datasets by making use of profiles. Even though this work seems promising, the results are only available on-demand. Moreover, the authors do not distribute their toolkit but rather a dataset as an example output of their approach.

The Flow-Level Anomaly Modeling Engine (FLAME) [17] tool is a promising work in this area. It works by taking as an input serial streams of flows and injecting hand-crafted network anomalies. As the name implies, this tool manipulates network flows. While this is useful in many circumstances, it also comes with a number of restrictions. First, datasets generated by FLAME cannot be utilized for evaluating intrusion detection algorithms in an agnostic manner; rather, they are limited to algorithms that use network flows. Lastly, FLAME, due to the fact that it is only working with network flows, is limited to the injection of attacks with flow footprints.

Summary

The topic of evaluating intrusion detection systems and algorithms exhibits various research gaps. Existing datasets are unable to provide a commonly accepted evaluation method due to their aforementioned disadvantages; they either contain undesired artifacts or they do not provide an holistic and up to date cover of cyber-attacks. Similarly, the related work for dynamically generating datasets has not been able to provide an holistic approach for such a task. Chapter 7 touches the topic of generating synthetic, yet realistic, intrusion detection datasets and attempts to address some of the problems identified in this section.

CONCLUSION

This chapter introduced the reader to a number of fundamental terms, definitions and related work. In more details, the introduction to IDSs served as background for the understanding of the next chapter, which tackles the topic of collaborative intrusion detection. Section's 2.2 introduction to honeypots, motivated their utilization both for the detection of attacks but also as an additional mechanism for alert data generation (see Part ii of the dissertation). Moreover, this section served as a discussion of the state of the art on honeypots with a focus on the ones intended for mobile systems and ICS environments. The identified research gaps are the basis on which Chapter 5 will built upon. The last part of this chapter discussed the topic of evaluating intrusion detection algorithms and systems. This topic is of high importance for any relevant future research, as no commonly accepted, generic and publicly available evaluation method exists. Chapter 7 copes with these challenges by proposing an approach for generating synthetic intrusion detection datasets. In a glance, the thesis at hand contributes in the fields of alert data generation and dataset creation in a threefold manner as depicted in Part ii and specifically in Chapters 5,6,7.

3

COLLABORATIVE INTRUSION DETECTION

This chapter is intended as an introduction to Collaborative IDSs (CIDSs) and to corresponding attacks on them. On the basis of the knowledge gained from the previous chapter it is now possible to dive into the topic of collaborative intrusion detection. First, Section 3.1 provides the reader with the basic background knowledge and motivation with regard to CIDSs. Section 3.2 proposes a number of functional and non-functional requirements towards the creation of a CIDS. These will be the basis for the evaluation of the state of the art (see Chapter 4) and for the development of the proposed CIDS in Chapter 9. Lastly, Section 3.3 discusses all aspects of attacks on CIDSs and concludes by connecting them with the aforementioned requirements. The thesis deals with such attacks on CIDSs in Chapter 10.

INTRODUCTION

The first IDSs have been mostly isolated single instances for monitoring a single system or a single network by carrying out local analysis for attacks. Hence, in between instances of such a stand-alone IDS, no communication and interaction takes place. Obviously, such a solution will not detect sophisticated and highly distributed attacks. That is, isolated IDSs will not be able to establish connections between malicious events occurring at different network places at the same time. Nowadays, with the increase in the size of corporate networks, malicious entities may attempt to spread their attacks so that their overall behavior can remain undetected. On the contrary, when collaboration of different IDSs would have been feasible, it would also be possible to perform alert data correlation and aggregation to detect adversaries that distribute their attacks all over the network.

Thus, for the protection of large networks and large IT ecosystems, *Collaborative IDSs (CIDSs)* emerged. CIDSs consist of several monitors that act as sensors and collect data. CIDSs can also assist to balance the load of IDSs in the cases of large-scale networks; isolated IDSs might not be able to efficiently monitor a big network due to the amount of traffic they would have to oversee. A CIDS can balance this task by distributing the effort to its monitors. Essentially, CIDSs enforce cooperation among different monitors and therefore they are more scalable than stand-alone IDSs. Besides improving the scalability and the detection accuracy of a monitored network, CIDSs can also significantly reduce the complex tasks of security administrators [65].

CIDSs usually contain one or several analysis units carrying out the actual intrusion detection on the data obtained from the monitors. Depending on the specifics of a CIDS, monitors and analysis units can be also co-located. If not indicated otherwise and for the remainder of this thesis, it is assumed that the term *monitoring sensor*[1] encompasses both a monitor and analysis unit.

CIDS can be roughly classified according to their communication architecture, as shown in Figure 4, into centralized, decentralized, and distributed CIDS:

- *Centralized CIDSs* consist of several monitors that observe the behavior of their respective host or the network traffic passing by. These monitors share their data with a *central analysis unit*. This data can be either alerts as a result of a local detection or extracted data from the local network traffic. Hence, the analysis unit is either applying alert correlation algorithms on top of received alerts or standard detection algorithms on top of the received network traffic data. Centralized CIDSs do not scale

1 Note that, for the sake of clarity, this thesis will be using the terms, *monitoring sensor* and *monitor* interchangeably.

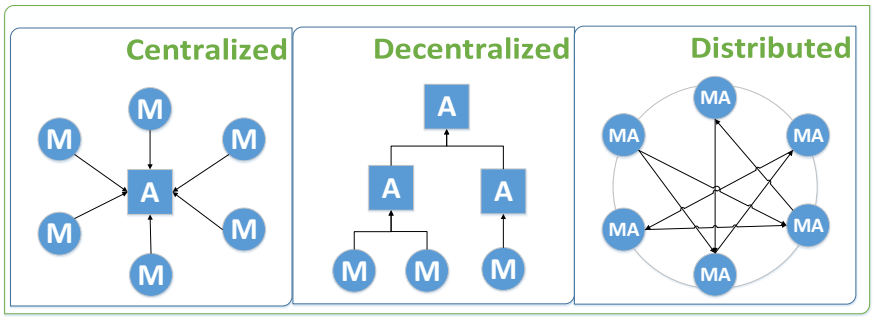


Figure 4: Overview of centralized, decentralized, and distributed IDS architectures that consist of monitors (*M*) and analysis units (*A*).

with the increasing size of the system that needs to be protected. Moreover, the central analysis unit represents a performance bottleneck and a Single Point of Failure (SPoF).

- *Decentralized (or hierarchical) CIDSs* usually make use of a hierarchical structure of monitoring points or multiple self-contained IDS deployments. Through this structure, they overcome the performance bottleneck of centralized CIDSs as they employ preprocessing and correlation of the monitored data within the hierarchy until the data converges to a central analysis unit on top.

- *Distributed CIDSs* share the tasks of the central analysis unit equally among all monitors, so that each monitor is also an analysis unit. Distributed CIDSs usually employ a P2P architecture, in which monitored data is correlated, aggregated, and analyzed in a completely distributed manner among the monitors.

Despite the fact that a lot of work has been done in surveying and classifying IDSs [38, 9, 25, 12], only a few focus specifically on CIDSs and their architecture [21, 196]. In particular, Zhou et al. [196] presented a study of CIDSs and focused mainly on the system architectures and the correlation of alert data. However, rather than surveying all related work in the field, Zhou et al. only shortly describe a few examples of existing CIDS proposals per architecture. This thesis additionally contributes in the area of CIDSs by more comprehensively discussing the state of the art in CIDSs in Chapter 4.

REQUIREMENTS

This section first proposes requirements of CIDSs for deployment in large networks and IT systems. Thereafter, the chapter discusses attacks on CIDSs and how they affect the requirements towards CIDSs.

The following functional and non-functional requirements for CIDS were identified in the course of this thesis:

- **High accuracy**: Accuracy for IDSs is determined by the percentage of successfully detected attacks and the corresponding percentage of undetected attacks (*false negatives*). In addition, the number of falsely triggered alarms (*false positives*) also needs to be taken into account to measure the accuracy of an IDS. An accurate IDS should minimize both.

- **Minimal overhead**: Overhead arises in terms of *computation* and *communication* effort. The techniques used to produce, collect, or correlate intrusion alerts must have a low computational overhead. In addition, the signaling inside the IDS, e.g., between monitors, or between monitors and an analysis units, needs to be as minimal as possible.

- **Scalability**: Scalability requires that the performance of the IDS increases linearly with the size of the resources added, so that networks of arbitrary size can be protected [69]. Therefore, the IDS should not contain bottlenecks or Single Point of Failures (SPoFs).

- **Resilience**: In the presence of failures on internal components and during attacks, a CIDS should still maintain its availability and ensure an acceptable accuracy. Hence, a CIDS should not only be resilient to system malfunctions, external attacks like Denial of Service (DoS) but also internal attacks from malicious CIDS components and malicious systems in the protected network/system. For this reason, a CIDS should prevent SPoFs and should provide graceful degradation and fast restoration mechanisms to counter failures and attacks.

- **Privacy protection**: In a collaborative environment, exchanged alerts may include sensitive information that needs to be protected and should not be shared or disclosed with all components in a CIDS. This is particularly important for CIDS deployments that share data across domains, which require privacy protection for the involved users, companies, and network providers.

- **Self-configuration** is the ability of the system to automatically adjust itself, without the intervention of an administrator. In contrast to systems that require manual configuration, this provides the ability of constructing less error-prone systems.

- **Interoperability** is the ability of the system to inter-operate with instances of the same system deployed in other networks, and also across different IDS implementations. For instance, this can be achieved, via the utilization of standardized formats such as the Intrusion Detection Message Exchange Format (IDMEF).

- **Domain awareness**: A real world deployment of a CIDS might be constrained by the existence of security policies. For instance, such a policy might forbid the communication of different sub-networks. As a result, CIDS sensors would not be able to properly communicate. In this context, domain awareness refers to the ability of the CIDS to, on-demand, constrain the dissemination of alert data to certain sub-domains of the monitored network.

ATTACKS ON CIDSS

CIDSs are essentially IT systems and thus they can be targets of attacks. Therefore, a comprehensive treatment of attacks has to include attacks on the CIDS components themselves.

Attacks on CIDSs can be classified into internal and external, based on the operating position of the attacker. External attacks refer to adversarial actions that originate from outside the monitored network. In an external attack, the adversary may try to detect the presence of a CIDS, launch evasion attacks (see Section 3.3.1.2), or attack specific components of the IDS directly, e.g., degrading its service availability via a DDoS attack [153, 111]. Internal attacks refer to malicious behavior originated from within the monitored network; either a host within the monitored network (network level) or a monitor that is part of the CIDS (monitor level) has been compromised. For example, via such an attack position, the malicious user can disclosure sensitive information (e.g., disclosure the CIDS's monitors) or aim on corrupt the alerts that are distributed between the CIDS monitors.

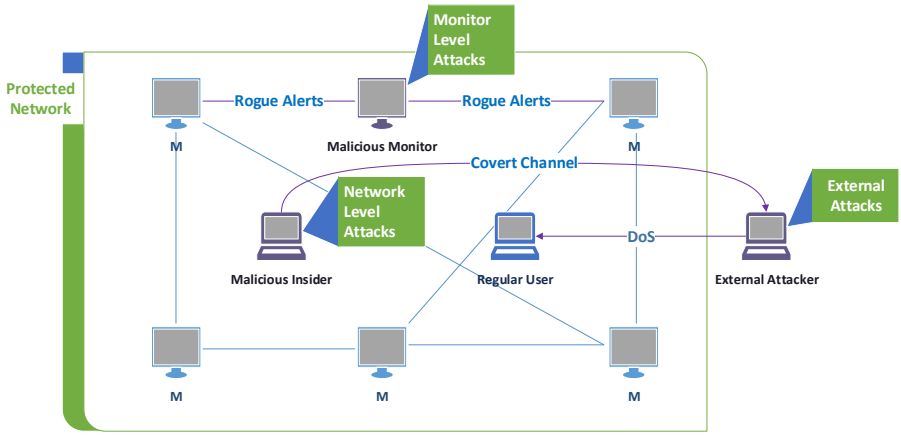


Figure 5: Possible network positions of attackers. M represents the different *monitoring points* of the CIDS.

The different positions of the adversary with respect to the aforementioned classifications are shown in Figure 5. This chapter considers three different network positions: an *external attacker*, e.g., carrying

out DDoS attacks, a *malicious insider*, e.g., performing covert channel attacks, and a *malicious monitoring point*, e.g., distributing fake alerts.

Figure 6 depicts an overview of attacks that can influence a CIDS with respect to the aforementioned internal and external level classification. The remaining of this section follows the detailed classification of Figure 6. External attacks can be further divided into the disclosure and evasion subclasses, while internal attacks are branched with respect to the position of the adversary to monitor-level and host-level.

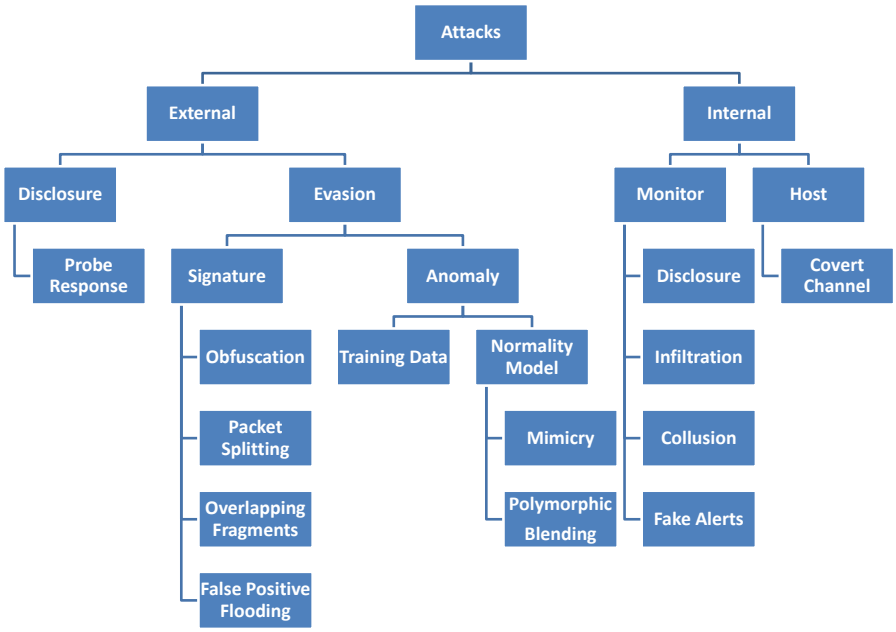


Figure 6: Overview of different attacks for CIDSs.

External attacks

External attacks have their origin outside the monitored network and can be classified into CIDS disclosure and evasion attacks. *Disclosure attacks* aim on detecting the presence of CIDS monitoring points in the network as preparation for subsequent *evasion attacks* to bypass the CIDS.

Disclosure Attacks

Disclosure attacks provide the attacker the means of identifying the monitors of a CIDS. In the following, this section briefly discusses the related work for such attacks. The reader can also refer to Chapter 10 which improves the state of the art as well as the respective mitigation mechanisms.

In [143], a method is presented for the discovery of passive monitors that publish their results publicly via the Internet. The assumption is that these results, e.g., periodically updated graphs that vi-

sualize the top attacked ports, provide enough information to trace the location of monitors. A similar but more active approach from Shmatikov et al. also assumes public CIDS output and introduces the concept of *Probe Response Attacks (PRAs)* [145]. The attacker carries out specifically adapted attacks, so that the produced alerts of the CIDS contain a unique marker. These markers are then used to identify monitoring points. In [13], passive sensor detection algorithms are presented that are based on PRAs.

Furthermore, Rajab et al. [129] describe techniques for live population sampling and methods for building sophisticated malware. In this case, the malware would try to spread intelligently over the Internet by firstly targeting only active IP space addresses and also by avoiding the disclosure by CIDSs. The first part is done on-the-fly via sampling techniques, in the different IP layers, accompanied by sending messages, e.g., ICMP packets. For avoiding disclosure, the assumption is that there is an increased probability that malware also attacks passive CIDS monitors while continuously choosing random IP addresses for further propagation. To prevent this, Rajab et al. propose the creation of malware with knowledge about the IP address ranges used by passive IDS monitors. Thus, such malware utilizes offline information regarding monitors that has been acquired via PRAs. In addition, the authors discuss the idea of malware that actively uses PRAs during their infection phase.

Disclosure techniques assume some kind of feedback path from the CIDS to the attacker, e.g., attack results that are visible on a public website. Hence, without such a feedback path, there are no methods for an adversary to successfully disclose monitoring points of a CIDSs. This thesis deals with PRAs by providing significant improvements to these attacks and their mitigation. For this, the reader can refer to Chapter 10.

Evasion Attacks

An evasion attack attempts to circumvent an IDS, so that no alert is triggered or to minimize the number of raised alerts. Depending on the applied detection method, a multitude of techniques are available for performing such an attack. With respect to the two major detection classes, evasion attacks can be differentiated into *signature* and *anomaly* based.

Signature-based evasion attacks try to evade an IDS that makes use of signatures for its attack detection. For that, an attack is modified, so that it does not match the known signature anymore. This requires to change the attack slightly, e.g., the order of events and packet payloads. Anomaly-based evasion attacks mainly focus on ways to masquerade an attack as legitimate and thus normal behavior.

SIGNATURE-BASED EVASION This class of evasion attacks usually attempts to take advantage of the static nature of signatures and the fact that even a small change (in a signature) can lead an IDSs to an oversight.

1. *Attack obfuscation* is one of the standard methods for an attacker to evade a signature-based IDS. The way to accomplish this is to transform its code into a semantically equivalent one that cannot be detected, as its signature is different compared to the origin code [26]. Depending on the level of mutation, an attacker may use either *payload mutation* where malicious payload packets are mutated to change their signature, or *shellcode mutation* in which a shellcode is obfuscated with polymorphic techniques. Several penetration testing tools support this feature, e.g., the Metasploit Framework[2].

2. *Packet splitting* is another evasion method [26] which exploits the fact that different operating systems handle fragmented packets distinctively. Hence, when the IDS cannot successfully reassemble fragmented IP packets or TCP segments, false negatives can be generated.

3. *Overlapping fragments* (or duplicate insertion attack) is a similar attack which is based thereupon that different operating systems handle overlapping IP fragments differently. When the IDS reassembles packets it may obtains different data than the target system of the attack, so that the attack is not detected.

4. *False positive flooding* can be finally launched to conceal an attack that is carried out in parallel., in the case when the signatures used by an IDS are known. This requires to create a large number of false positives at the IDS, e.g., via available attacking tools like *Inundator*[3], *rule2alert* [4], or *idswakeup*[5].

ANOMALY-BASED EVASION Anomaly-based detection techniques [25] create a model of the normal traffic, and consider any deviation as a possible attack (see Chapter 2.1). In most cases a training phase is required to determine the normal behavior. As mentioned in Chapter 2 and in contrast to signature-based detection, the main advantage of using anomaly detection is the ability to detect unknown attacks [9]. However, this usually comes at the cost of an increased false positive ratio. There are two possible ways for an attacker to evade an anomaly detection system. The attacker can either modify what is considered to be normal (*training data* attack) or the attack can be transformed to appear as normal behavior (*normality model* attack) [159].

2 http://www.metasploit.com
3 http://inundator.sourceforge.net
4 http://code.google.com/p/rule2alert
5 http://www.hsc.fr/ressources/outils/idswakeup

1. *Injecting training data attack* refers to the case where an adversary is able to inject intrusive behavior during the training period of an anomaly-based IDS detection algorithm. Subsequently, the system will not be able to distinguish such an attack from normal behavior in the future. Moreover, as the overall behavior changes over time, periodic training, while the system is active, is required [38]. This can be exploited by attackers to incorporate intrusive behavior patterns to the training data of the anomaly detection.

2. *Mimicry attacks*, a subclass of anomaly-based evasion attacks, was introduced by Wagner et al. [184]. The authors introduce evasion techniques that are capable of bypassing anomaly-based IDS. This means that an attack is transformed in a way that seems legitimate by imitating normal activity. Tan et al. also confirmed that mimicry attacks are a real threat for anomaly detection by using modified real-world exploits to evade IDSs [159]. The main idea behind the aforementioned attacks is to insert dummy system calls into the attack sequence, so that the final overall system call sequence looks normal. Moreover, Kruegel et al. [81] expanded and automated this class of attacks, resulting in the successful evasion of anomaly-based IDSs.

3. *Polymorphic blending attacks*, which are similar to mimicry attacks, were introduced in [51]. The assumption behind such an attack is that only a small part of normal traffic can be analyzed by an anomaly detection algorithm, because of difficulties arising with the modeling of complex systems and also due to performance overhead issues. To exploit this, Fogla et al. propose a combination of mimicry techniques with polymorphism, which introduce the ability of changing the appearance of an attack with every instance [51]. For that, the attacker first creates a profile of what is supposed to be normal traffic by the IDS and then encrypts the attack body to blend it with the learned profile. Finally, a polymorphic decryptor is generated, to decrypt the attack body when needed. Authors demonstrate the feasibility of the proposed attack via attacking the *PAYL* anomaly-based IDS, which is specialized in detecting polymorphic attacks [188, 187].

Internal attacks

Internal attacks refer to malicious behavior originated from within the monitored network. An insider can be classified as either a *malicious monitor* that is *part* of the CIDS monitoring topology (monitoring level) or a *malicious host* inside the monitored network (see Figure 5).

Compromised host

Once a host is compromised within the protected network, a covert channel can be set up between this host and an external entity. A covert channel tries to hide the very existence of any communication [85, 191] by using a channel that is usually not intended for communication, e.g., timing information in between packets or unused bits in the IP header. Covert channels presume a compromised host within the protected network, and depending on the specific channel used, they can theoretically evade any CIDS.

Compromised monitor

It gets even worse, when the attacker has compromised a component of the CIDS, e.g., a CIDS monitor, as this enables the adversary to launch subsequent attacks: when the attacker has successfully infiltrated a CIDS, other monitoring sensors can be disclosed. Furthermore, a malicious user can compromise additional CIDS components or exploit vulnerabilities in the CIDS protocol to let compromised components take over more important positions or functions in the CIDS overlay, e.g., by producing fake alerts, accusing other monitors to be compromised or by conducting supporting DoS attacks on other sensors. In addition, multiple compromised monitoring sensors can collude to increase the chances of taking down or compromising the rest of the monitored network [53]. Moreover, on the basis of a compromised monitor, the attacker can easily bypass the CIDS and thus reduce its accuracy by selectively forwarding alerts to other monitors, e.g., by suppressing alerts for specific attacks.

Recent and more sophisticated attacks attempt to circumvent a common countermeasure to internal attacks, namely the adoption of reputation systems [131, 105], e.g., EigenTrust [73]. In such an adoption, each of the monitors establishes a certain trust level based on its detection behavior or other defined properties. Whenever the trust level drops below a specified threshold, the monitor is considered non-trustworthy and specific measures can take place, e.g., blacklisting of the specific monitor. Specifically adapted to CIDSs, Fung et al. [55] propose a trust management model that is based on Bayesian probabilities. In more details, when monitoring points distribute their alert data, they also send request messages to determine the trustworthiness of other monitors. This is achieved by a probabilistic model whose purpose is to measure the satisfaction level of the received response messages. A number of similar trust mechanisms for CIDSs to cope with insider attacks have been proposed [54, 41, 141, 62].

While the usage of such mechanisms protects the system from many of the aforementioned attacks, it creates new opportunities for the attacker. A well known problem with reputation systems is the exploitation of the system itself when highly trustworthy peer(s) is

compromised, which is called a *betrayal attack* [54]. If the trust value of the compromised peer is not quickly degraded, the overall accuracy of the system will be affected. In another variant, a so-called *sleeper attack* [18], a malicious peer first behaves benign over time to establish a certain reputation level before it carries out the actual attack.

The topology of a CIDS might be completely static and pre-configured or when monitors are added to the system dynamically, a strict access control for them can be enforced. However, in the highly unlikely case of a CIDS that is open and allows the dynamic inclusion of additional monitors, an attacker can launch a *sybil attack* [40] by adding a multitude of malicious monitors to the system [53]. These can be used to establish a more detailed view of the CIDS topology and to prepare subsequent attacks, e.g., to degrade the detection accuracy of the CIDS, to out-vote honest nodes, to perform whitewashing of malicious peers, and to compromise additional monitors. However, such attacks could be considered rare as most CIDS topologies are usually rather static and CIDSs may enforce strict authentication mechanisms for new monitoring sensors.

Discussion

This chapter has described external and internal attacks on CIDSs. External attacks can disclose the presence of a CIDS and decrease its detection accuracy by evasion attacks. An attacker who successfully compromises a host in the protected network can launch a multitude of additional attacks, e.g., setting up a covert channel to evade the IDS and thus decreasing its detection accuracy, e.g., to hide an export of sensitive data. It gets worse, when the attacker compromises a CIDS monitor, as it allows to bypass the whole CIDS, degrade its detection accuracy, or in worst-case to bring it down completely.

Internal attacks are more effective when combined with external ones. Thus, compromised hosts or monitors may provide information to the outside, which is used to prepare subsequent external attacks. For instance, a sophisticated attack may include an evasion technique to compromise a peer inside the monitored network without triggering any alerts. Afterwards, the adversary could use covert channels [191] to send sensitive data outside the protected network.

Table 1 summarizes the aforementioned attacks with respect to the requirements given in Section 3.2. Any type of attack on a CIDS is also an attack on its main task, namely the detection of attacks on the protected network and thus an attempt to degrade the detection *accuracy* of the respective CIDS. *Overhead* issues could arise through malicious monitors, DoS attacks or even network-based insider attacks. For instance, an adversary who controls a CIDS monitor can flood the CIDS network. *Scalability* is mostly an architectural property and there-

CIDS Attacks	*Highy Acc.*	*Min. Overh.*	*Scalability*	*Resilience*	*Priv. Prot.*	*Self-config.*	*Interoper.*	*Dom. Awar.*
External/Evasion/Signatures	✓	×	×	✓	×	×	×	×
External/Evasion/Anomaly	✓	×	×	✓	×	×	×	×
External/Disclosure/Probe Response	✓	×	×	×	×	×	×	×
Internal/Monitor	✓	✓	×	✓	✓	×	×	✓
Internal/Host	✓	✓	×	✓	[✓]	×	×	×

Table 1: Relationship between Attacks on CIDSs and Requirements: Check marks ✓ indicate a relation between an individual requirement and IDS attacks, while checkmark symbols in brackets [✓] an indirect relation. Finally, x marks ✗ show the absence of any relationship.

fore it is not affected by the aforementioned attacks[6]. Furthermore, *resilience* is related with most of the CIDS attacks (e.g., internal attacks). Attacks on the *privacy protection* of CIDSs are mainly related to malicious monitoring points as the exchanged CIDS alert data may contain sensitive information. For instance, this could be the case when several organizations use a single and interconnected CIDS. However, insiders may also, indirectly, affect the *privacy* of the involved participants as they can disclose sensitive data to unauthorized external parties. The *domain awareness* property can be affected in the case of compromised monitors. For example, in such an event the adversary might attempt to disregard the CIDS protocol and contact network domains that are considered restricted. Lastly, *interoperability* and *self-configuration* are non-functional requirements and hence do not directly relate with the aforementioned attacks.

6 However, scalability can be affected by DoS/DDoS attacks, in the context of the system not being able to support additional monitors during such an attack.

4

TAXONOMY AND STATE-OF-THE-ART

The previous chapters aimed at supporting the reader by presenting all the fundamental background work and the necessary definitions that will be used in the course of the thesis. This chapter provides with the first original contribution by proposing a novel taxonomy for collaborative intrusion detection (Section 4.1). The taxonomy is described in-depth and it is followed by a comprehensive survey and qualitative comparison of the state of the art (Section 4.2). The current chapter drives the thesis forward in a twofold manner. First, it proposes a systematic approach for designing a CIDS and also examines the majority of the related work. Second, the survey of the state of the art reveals the research challenges that are yet to be addressed.

TAXONOMY OF COLLABORATIVE INTRUSION DETECTION

The challenge of fulfilling all requirements put forth in Chapter 3.2, calls for a systematic approach which in turn calls for a decomposition of concerns. In the process, a number of challenges arise, e.g., minimizing the exchanged data by maximizing the detection accuracy, deciding which monitors should exchange information, and identifying the most efficient membership management architecture for the monitors.

To structure the solution space for such a system, a disjunction of CIDSs into five main building blocks is proposed (see Figure 7). The suggested separation attempts to create a logical guide, for researchers, which takes into account all major steps that are necessary for the monitoring and identification of targeted and sophisticated attacks in large networks.

Local monitoring stands for all the monitor-level detection mechanisms that are deployed in the CIDS (e.g., a honeypot or a signature-based IDS). The *Membership management* is related to the task of ensuring the overall connectivity of the monitors that assemble the CIDS. Moreover, the *data dissemination* relates to all the mechanisms that the CIDS is utilizing to exchange alert data in-between different monitoring sensors. *Correlation and aggregation* can be performed both locally (in each monitor) but also in the whole CIDS and can significantly benefit the overall detection accuracy of the system. Lastly, the *global monitoring* block refers to the ability of the CIDS to detect attacks as a result of its internal collaboration and correlation of alert data.

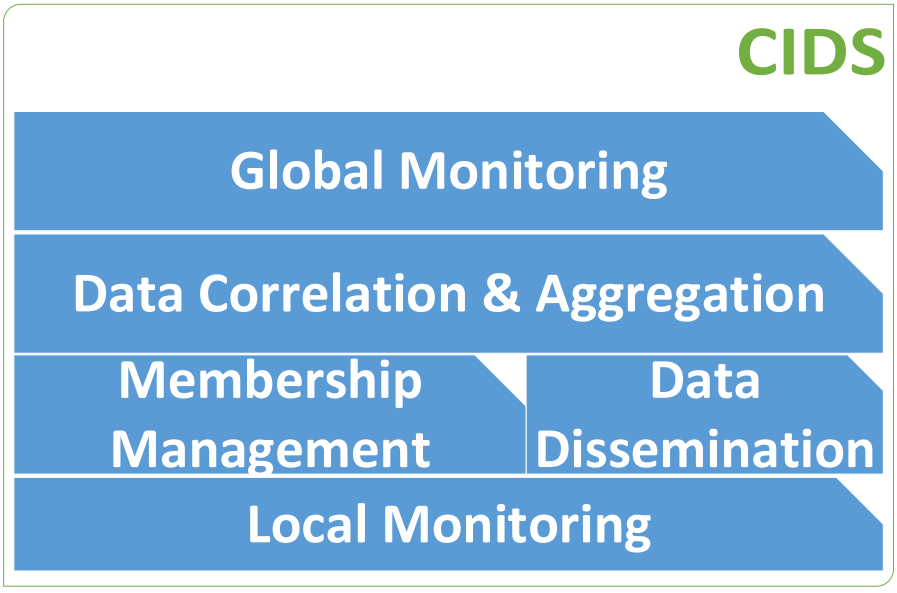


Figure 7: Building blocks for CIDSs.

The aforementioned building blocks lead to a detailed taxonomy that is proposed in this thesis, as shown in Figure 8. In the following,

details are given for each building block and its corresponding design space.

Local monitoring

Local monitoring in a CIDS can take place either on *host* or *network level*. On the host level, this requires the monitoring of local activities to identify malicious behavior, which presumes monitoring functionality on all hosts of the network. In contrast, by monitoring at the *network level*, an entire network can be protected by deploying monitoring points only at strategically selected network locations, e.g., close to the ingress and egress routers. Combinations of host and network level monitoring are also feasible and will increase the amount of monitored data, thus allowing for a more fine-grained attack detection in a CIDS.

Monitoring is classified as either *passive* or *active*. Passive monitoring corresponds to scanning local activity or the locally observed network traffic. In active monitoring, *honeypots* can be used to emulate the presence of vulnerable systems as promising attack targets. As they have no productivity use, any interaction with them can be classified as an attack (see Chapter 2.2). Hence, honeypots produce no false positives, although their false negative rate can be high.

In the context of local monitoring, detection engines are the individual mechanisms used in the analysis units of a CIDS to detect attacks in the data collected by passive sensors. As discussed in Chapter 2, besides honeypots, such detection is either *signature-based* or *anomaly-based*, however combinations of both mechanisms are also possible. For instance, Bro IDS [120] contains, beyond signature-based, modules that can be utilized for anomaly detection. Signature-based detection requires existing signatures for an attack and thus is unable to detect unknown attacks. Anomaly-based detection requires to create a model of the normal behavior of the system. Each deviation from this model is then interpreted as an anomaly and thus as an attack. Hence, an anomaly-based detection can also detect unknown attacks. A comprehensive survey of anomaly-based detection methods is given in [25] and [59].

To sum up, in an optimal CIDS design, the local monitoring should be agnostic of the specifics of the utilized detection mechanisms. In this sense, monitors in such a system can be seen as a means for generating alert data that will be exchanged via the utilization of the other blocks of the CIDS. Such a task can be envisioned by supporting interoperable alert data exchange formats, e.g., the IDMEF (cf. Chapter 3.2).

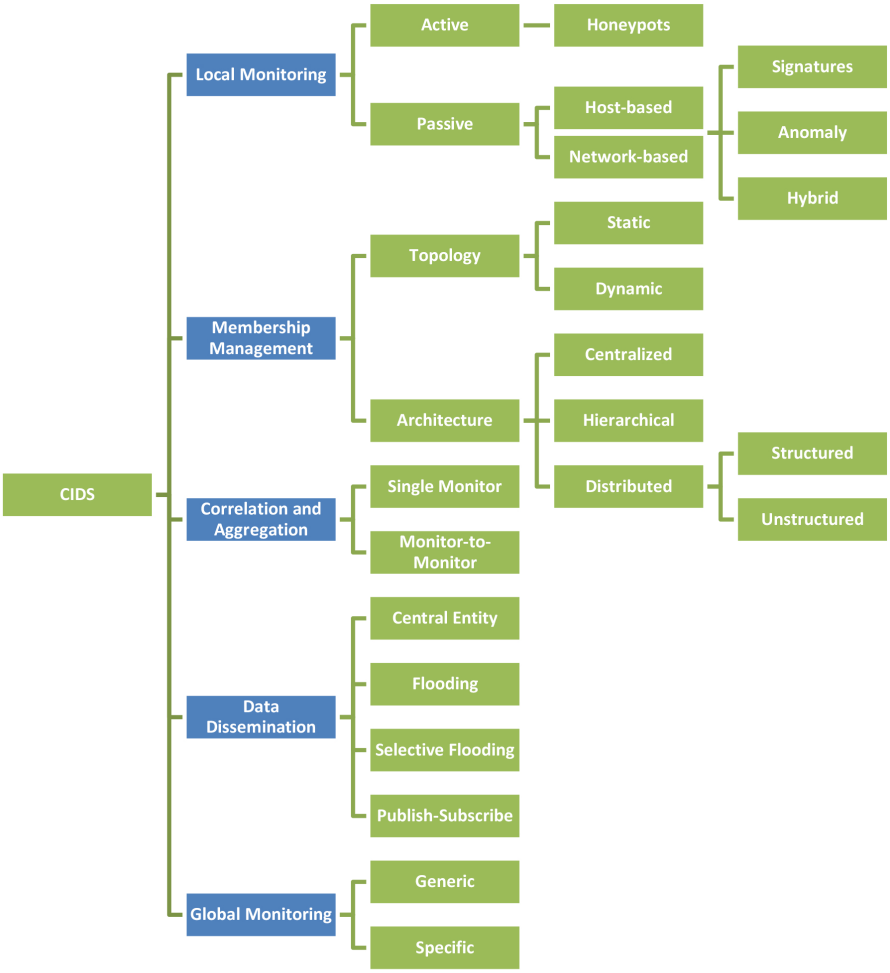


Figure 8: Taxonomy of CIDSs.

Membership management

The membership management component of a CIDS is responsible for the task of ensuring the overall connectivity of the monitoring overlay of the CIDS, by managing the neighborhood relations between monitors. Depending on the CIDS, such a membership management can result in static or more dynamic overlays that allow the dynamic inclusion and exclusion of monitors.

In the simplest case, the connections in the CIDS overlay are static and pre-determined. Hence, an administrator needs to be involved whenever new components are added to the system. Alternatively, the CIDS overlay is set up dynamically. This can be done either via a central server that features a global view of the system or via a membership management protocol that runs at each monitor and which operates on local knowledge only. This classification of *static* and *dynamic* overlays is also shown in Figure 8. As it will become evident in

Section 4.2.4, the majority of CIDSs adopt a dynamic overlay architecture.

As the membership management controls the overlay neighborhood of CIDS components, it can also enforce a certain network architectural structure on it. Hence, the monitoring overlay can be either centralized, hierarchical, or completely distributed. In a *centralized CIDS*, all monitors are directly connected to a central analysis unit. A *hierarchical (or decentralized) CIDS* arranges all monitoring sensors in a hierarchy that is rooted at a central analysis unit. Hence, monitors in lower tree levels report to the monitors of higher levels. In addition, *hierarchical CIDSs* include approaches that make use of a number of supernodes. Such an architecture retains the hierarchical structure but also provides certain nodes additional privileges, e.g., the ability to correlate data.

Distributed CIDSs prevent any SPoFs by deploying monitors in a flat overlay without exposed components like a central analysis unit. Furthermore, the membership management can either exhibit an *unstructured* overlay ID space or it can enforce a *structure*, in case an additional location service is required, e.g., on the basis of a Distributed Hash Table (DHT) [4]. Thus, a structured ID space would provide the advantage of a guaranteed broadcast and search functionality. However, for CIDSs this requires to map monitored data to a one-dimensional ID space. Therefore, multi-dimensional alert correlation (cf. Section 4.1.3) cannot easily be achieved by structured CIDS overlays. Depending on the observed attack scenario, selecting the right pattern as key for the DHT is crucial. For instance, a CIDS whose purpose is to be able to detect attacks originating from the same source, is usually making use of the source addresses from the monitored packets as the DHT key.

Correlation and aggregation

Once data has been obtained and analyzed by the local monitoring block, possible alerts need to be correlated and the monitored data needs to be aggregated for a later dissemination to other monitors or analysis units (cf. Section 4.1.4).

We distinguish between *single-monitor* and *monitor-to-monitor* correlation mechanisms. *Single-monitor* correlation correlates alerts/data locally at each monitor without sharing this information with other monitors. Hence, plain alerts or locally correlated alerts are shared either directly with an administrative interface of the CIDS or with a central analysis unit that carries out further correlation. *Monitor-to-monitor* correlation enforces the sharing of alerts/data with other monitors that will attempt to correlate this information with local information. Therefore, such a correlation technique requires to share alerts or even more detailed data with other monitors. For this rea-

son, sharing blacklists of malicious IP addresses still remains a *single-monitor* correlation approach as this information cannot be seen as an explicit input to the local correlation with local data.

Resulting and correlated alert patterns can have multiple dimensions, which is a major challenge for collaborative intrusion detection [179]. For example, to detect an attack that is conducted from several source nodes simultaneously, it is not sufficient to correlate alerts solely based upon the IP addresses of the attack sources. Moreover, if more than one system is attacked at the same time, an alert correlation via the IP addresses of the victims is also not meaningful. Hence, a plethora of different patterns and combinations of them are imaginable for alert correlation. For instance, combinations of source IP, destination IP, protocol, source port, destination port, and even payloads of monitored packets can be used.

Alert correlation techniques, in general, can be classified into the following four different approaches [196, 47]:

- *Similarity-based correlation* approaches, e.g., [166, 37, 33], correlate alerts based upon the similarity of data or alert attributes. The similarity of two data sets is reflected by a score that is computed by similarity functions. Depending on the produced score, the data is then either correlated or not.

- *Attack scenario-based* approaches take causality into account when correlating data/alerts. Thus, they allow detecting complex attacks that take place in several steps. Such approaches, e.g., [36, 58, 44], usually require to establish an attack database. Furthermore, most of these approaches need to be initialized by a training data set. Therefore, they provide high accuracy for known attacks, but fail in detecting unknown attacks.

- *Multi-stage alert correlation* techniques aim at detecting unknown multi-step attacks. Most such approaches presume the existence of relations among the different stages of an attack. Thus, they presume that an attack is conducted to prepare another one [174]. Multi-stage alert correlation usually requires building up a library of attack steps. Depending on the overall attack, multiple steps are then mapped and/or correlated to attack scenarios.

- *Filter-based* approaches, e.g., [125], attempt to filter irrelevant data or alerts to reduce the number of false positives in CIDSs. For that, alerts are prioritized according to their impact on the protected system. Thus, such approaches require a detailed description of the system to be protected, e.g., its network topology and the deployed operating systems, that is not always available. Moreover, the accuracy of the alert correlation depends on the level of detail provided in the system description.

Data dissemination

Correlated alerts and aggregated data need to be efficiently distributed to avoid unnecessary overhead in a CIDS. The data to be disseminated can range from alerts to monitored data at all possible kinds of aggregation granularity. Especially CIDSs that focus on the detection of highly tailored and targeted attacks require data sharing beyond simple alert dissemination.

The data dissemination is heavily influenced by the CIDS architecture and therefore by the applied membership management.

CENTRALIZED AND DECENTRALIZED CIDSS On the one hand, centralized CIDS have a pre-defined and directed flow of information, namely from monitors to the analysis unit. On the other hand, decentralized CIDSs arrange their monitors in a hierarchy with a strict bottom-up flow of information. This hierarchy can be either completely static or changing dynamically, e.g., on the basis of the monitored data.

DISTRIBUTED CIDSS In contrast, distributed CIDSs provide a flat monitoring overlay and the highest level of freedom in exchanging data in between monitors. Data dissemination in distributed CIDS can either result in flooding the entire CIDS overlay or in a *selective/partial flooding* by random walks [181] or gossiping approaches [57]. A selective, yet more intelligent flooding of a distributed CIDS overlay can be provided by using *publish-subscribe* methods. In such a scenario monitors subscribe to other monitors for specific information, e.g., alert data. This establishes groups of monitors that are interested in the same kind of information. Within such groups, data can be exchanged in both directions and thus it is not limited to the direction from publisher to subscribers. In distributed and DHT-based CIDSs, subscribers can send information to the respective publishers, e.g., via a reverse multicast on top of DHT-based publish-subscribe.

Global Monitoring

For the detection of distributed attacks, a global monitoring mechanism is required that is built upon collaboration and information exchange between monitors. This global monitoring, which is based upon the data correlation and aggregation from Section 4.1.3, represents the detection capabilities of the respective CIDS.

Depending on the detection scope of the CIDS, the global monitoring can vary from being *generic* to *specific*. Generic global monitoring indicates systems that attempt to detect as many attacks as possible without having any specialization to any specific attack class. The specific scope, in the context of CIDSs, can refer to various specialized

detection classes. In particular, CIDSs may concentrate on detecting insider attacks, DDoS attacks, on identifying malware that is attempting to spread over the monitored network, etc. In a glance, the global monitoring building block refers to the collaborative capabilities of a CIDS as well as to its detection scope.

The remainder of this chapter contains a detailed survey and comparison of existing CIDS systems, starting with an overview of centralized CIDSs in the subsequent section.

STATE-OF-THE-ART

In the following a comprehensive analysis of the existing work in CIDSs is given. For this, we utilize the membership management building block (cf. Section 4.1.2) as the basis for categorizing the systems into the centralized, hierarchical and distributed classes.

Centralized CIDSs

In a centralized CIDS, monitors send all their information directly to a central analysis unit that either applies detection algorithms and/or alert correlation algorithms on the overall data. These systems are widely used as they provide high accuracy rates at low architectural complexity. However, in most centralized CIDSs, monitors are usually configured manually. Furthermore, such systems do not scale with the number of monitors and thus cannot protect large networks. In addition, as all data is collected and all analysis is done at one central unit, they might not be applicable for collaboration across different organizations due to privacy issues.

DIDS

Snapp et al. proposed the Distributed Intrusion Detection System (DIDS) [146] as one of the earliest centralized CIDSs in literature. DIDS attempts to detect malicious activity, over the monitored network, and create an overall score of its security state. The DIDS architecture combines distributed monitoring with a centralized data analysis. DIDS consists of three basic components: the *DIDS director* that represents a central analysis unit, *host monitors*, and *network monitors*.

OVERVIEW Network monitors, observe all packets that are transmitted in their observed network segment. They apply simple host analysis techniques, e.g., monitoring of certain services and protocols such as *rlogin* and *telnet*, and utilize heuristics to identify potentially intrusive behavior. The host monitor is responsible, for the monitoring of a particular host. This unit also conducts a preliminary event analysis to decide which of the alerts should be forwarded to the

director. The existence of a host monitor is not mandatory, as the network monitor can also report network activities of hosts. In addition, in both monitor levels aggregation is done by removing non-significant or OS-specific data before sending them to the central analysis unit. The main component of DIDS is the director, a centralized expert system that receives all alerts from host and network monitors. At this point data is aggregated via a rule-based expert system and analyzed. Afterwards, the system decides whether there is a security breach on a certain host or a large-scale attack on the whole system. Finally, some correlation techniques are applied in DIDS. For instance, the system creates a unique ID for each monitor entering the monitoring environment. Subsequently, any malicious activity related with this particular ID is consider part of the same attack.

REQUIREMENTS DIDS applies only simplistic detection techniques that can be evaded by a sophisticated adversary. Hence, the accuracy of the system can be rather poor. Moreover, another disadvantage of this CIDS is the lack of self-configuration mechanisms. Furthermore, as the communication and computation overhead at the director increases with an increasing size of the monitored network, DIDS does not scale. The director component in DIDS is a SPoF and thus violates the *Resilience* requirement.

SURFcert IDS

SURFcert IDS[1] is a centralized CIDS, that is based solely on honeypots. *SURFcert's IDS* main scope is to create a large-scale CIDS that exhibits zero (or a really low ratio of) false positives.

OVERVIEW The system comprises of multiple monitoring points, so-called *passive sensors*, that forward all their traffic via pre-established Virtual Private Network (VPN) tunnels to a centralized analysis unit, the so-called *tunnel/honeypot* server. At the tunnel server, the traffic is then analyzed by one or more honeypots and the results are stored on a separate logging server.

REQUIREMENTS In terms of accuracy, the exclusive usage of honeypots results in a zero false positive rate, as any interaction with these systems is considered to be an attack. Nevertheless, honeypots cannot detect all attacks as they presume an interaction of the attacker with the honeypot. SURFcert IDS uses *Nepenthes* [10], its successor *Dionaea*[2], the *Kippo*[3] SSH honeypot, and *Argos* [126] as a secure system emulator. All of these honeypots can only emulate certain ports and protocols. Hence, they cannot cover all possible attacks. This also

1 http://ids.surfnet.nl
2 http://dionaea.carnivore.it
3 http://code.google.com/p/kippo

illustrates why honeypots should not be used as the only detection method in an IDSs, but rather as an additional detection technology. Moreover, both computational and communication overhead of the tunnel server increase proportionally with the increasing number of sensors. Thus, the tunnel/honeypot server is not only a SPoF but also rendering the system to be not scalable. In addition, there is an absence of alert correlation and aggregation mechanisms, as all alert data is transferred to the central analysis unit. Finally, *SURFcert IDS* does not provide significant global monitoring capabilities, as it is only able to present an overview of the local detected attacks (along with some statistics).

CRIM

Cooperative Intrusion Detection Framework (CRIM), introduced in [34], is a centralized cooperative module that obtains data from monitors or rather isolated IDSs. CRIM's scope is to analyze alerts and subsequently attempt to identify the adversaries' next possible steps. It provides functions for managing, clustering, merging, and correlating alerts and thus takes over the task of a central analysis unit in a CIDS. Monitors send their alerts in the standardized IDMEF data format [39] to the CRIM module.

OVERVIEW To process the received data, an alert management function converts data to a set of tuples, which are then saved in a relational database. Afterwards, an alert clustering function generates clusters of alerts based on a relation of similarity [33]. The similarity relation between two alerts is created by an expert system and it is based on the classification of the alert, time, source, and target. Clusters are inserted to an alert merging function that creates new global alerts. The global alerts, consist of the alert data collected from each cluster. Subsequently, global alerts feed a correlation function which conducts further analysis and creates a set of possible actions that might be performed by the adversary based on the current alert data. Finally, an intention recognition function is used to provide the administrator with attack information and the possible next steps of the attacker.

REQUIREMENTS On the one hand, much of the authors' work is focused on correlation methods, which seriously reduces the overall overhead. On the other hand, the multiple merging and correlation that is performed by several functions, may lead to excessively abstract alerts creating a limited detection coverage, and thus a poor accuracy. Furthermore, the proposed similarity-based correlation mechanism will not be able to relate sophisticated attacks. For instance, a slow distributed attack, from different sources, to different parts of the monitored network would remain undetected. Finally, the usage

of IDMEF as a standardized language for exchanging alerts is an advantage in terms of the system's interoperability.

DIDMA

Distributed Intrusion Detection system using Mobile Agents (DIDMA) [74] is a CIDS for the detection of distributed attacks on large networks. DIDMA makes use of static agents, that act as local monitors. In addition, mobile agents exist, that are responsible for alert dissemination as well as correlation and aggregation of data. Moreover, DIDMA contains a centralized entity that maintains lists of hosts experiencing similar attacks.

OVERVIEW A DIDMA network can be seen as a static overlay in which local agents communicate with mobile agents. These local agents act as host monitors and generate alerts whenever malicious activity is detected, which also includes a classification of the type of the detected attack. Based on this classification, a global list of IP addresses is kept for nodes affected by the same type of attack, e.g., a DoS attack. Whenever an alert is generated by an agent, the IP address of the respective host is added in the list. Upon the occurrence of malicious activity and to detect a possible intrusion in the network, a central entity creates a mobile agent that can be transferred to other network positions. For each identified attack, the mobile agent updates the global list with IP addresses of other hosts that exhibit similar suspicious activity. A mobile agent is then sent out and subsequently visits all hosts that are listed for the same type of attack. During this process, the mobile agent aggregates and correlates information from the visited hosts, updates the global list, and generates alerts when detecting a specific attack. In the end, alerts are sent to a central user interface for further analysis.

REQUIREMENTS DIDMA utilizes a central entity for creating a global view of the overlay, as well as for dispatching mobile agents, that represents a SPoF. DIDMA utilizes only a signature-based detection algorithm and therefore cannot detect unknown attacks. Furthermore, the system requires a valid classification of the alerts, for the agents to operate properly. As this is not always possible the overall accuracy could be low. The system's overhead is highly affected by the number of hosts that are under attack, which corresponds to the number of hosts added to the global list. In addition, overhead arises when a high percentage of the detected attacks cannot be aggregated. In both of these worst-case scenarios DIDMA may produce considerably high communication overhead. DIDMA's resilience can be seriously affected in the case of malicious agents. Furthermore, if a host is compromised an adversary could also compromise mobile agents.

Summary

Centralized IDSs usually provide high detection accuracy rates. Their main disadvantages are the lack of scalability in terms of the number of supported monitors and the SPoFs that is posed by the central analysis unit. Nevertheless, due to their higher accuracy, centralized IDSs are widely used in small to medium-sized corporate networks. DIDS [146] was one of the first centralized approaches, and many systems followed its basic architecture. In addition, *CRIM* [34] focuses on the correlation and aggregation of alert data. More recent approaches like *SURFcert* IDS provide interesting enhancements, e.g., honeypots as an additional detection mechanism, while others, e.g., DIDMA, make use of mobile agents. Finally, Table 2 provides a summary of centralized CIDSs and their main building blocks (for an overall comparison of all surveyed CIDSs the reader can refer to Tables 5 and 6).

CIDS	*Local Monitoring*	*Correlation & Aggregation*	*Membership Management*	*Data Dissemination*	*Global Monitoring*
DIDS	Host-based	Single Monitor	Centralized	Central Entity	Generic
SURFcert	Honeypots	✖	Centralized	Central Entity	Generic
CRIM	Network-based	Single Monitor	Centralized	Central Entity	Attack Prediction
DIDMA	Network-based, Signatures	Monitor-to-Monitor, Similarity	Centralized, Static	Selective Flooding	Generic

Table 2: Centralized CIDSs and their Building Blocks. *The x marks ✗ indicate that the respective building block is not available.*

Hierarchical CIDSs

Decentralized CIDSs organize monitoring points, or several different self-contained IDS deployments, hierarchically in a tree that is rooted at a central analysis unit. Within the tree, preprocessing and correlation of the monitored data takes place. On the basis of this correlated data, a distributed analysis for the detection of attacks takes place. However, when data aggregation in a hierarchy takes place, then information is lost in each level of the hierarchy. As a result, highly distributed and sophisticated attacks may remain undetected.

GrIDS

The *Graph Based Intrusion Detection System (GrIDS)* is intended for the protection of large networks from actively propagating malware [155, 28], but can also detect attacks on individual hosts.

OVERVIEW The network is split into several zones, called departments, that are organized in a tree-like structure. Each department contains one analysis unit and several network and host monitors that perform intrusion detection and subsequently send their data

to the analysis unit. Hence, each host in GrIDS belongs to a department, while the departments are controlled by parental departments, thus creating a hierarchy. Moreover, each department contains two special modules: a *software manager* and a *graph engine*. The software manager is responsible for the management of the local hierarchy status, as well as the monitors within a department. The overall tree hierarchy is ensured by a centralized hierarchy server. Finally, GrIDS provides the ability to make dynamic changes in the hierarchy, via the utilization of a user interface.

The graph engine receives input from monitors within a department and thereupon establishes activity graphs that represent hosts and the network activities between each other. These activity graphs are firstly analyzed locally, and afterwards, they are aggregated and passed upwards to the parental department and its graph engine. At this point, all information from child departments is merged and graphs with coarser resolution are established. Suspicious behavior is detected on the basis of user-given detection rules that are expressed through a defined policy language.

REQUIREMENTS The usage of detection rules suggests that in terms of accuracy, the system would only detect attacks that are a violation of a security policy. Therefore, sophisticated or unknown attacks may still remain undetected. In addition, the aggregation mechanism may not be able to detect a widespread attack that progresses slowly, as only attacks that occur within a short time period can be detected. Due to the division of the overall network into departments and their hierarchical organization, GrIDS is scalable in terms of protecting networks of arbitrary size. However, the hierarchy server that controls and maintains this hierarchy is a SPoF and serves as a potential bottleneck. GrIDS is vulnerable to DoS and insider attacks, as with most of the centralized and decentralized CIDSs discussed in this chapter. Finally, the system exhibits a built-in privacy-protection mechanism due to the way it handles its hierarchy; each department is only able to observe activity that is restricted within its boundaries.

AAFID

The *Autonomous Agents For Intrusion Detection (AAFID)* is a hierarchical CIDS proposed in [11, 149]. AAFID does not focus on the detection of specific types of attacks but rather acts as a framework in which different detection engines may be utilized. The system consists of *agents*, *transceivers*, and *monitors*. With respect to our terminology, agents act as monitors, while transceivers and monitors act as analysis units.

OVERVIEW *Agents* are stationary in the AAFID architecture and it is not foreseen that they migrate between different hosts. Each host can contain multiple agents that perform event monitoring and af-

terwards send their reports to a transceiver. For instance, an agent could monitor for large number of port scans targeting a protected host. As soon as it detects this kind of activity, it will generate a report and send it to a transceiver. The authors claim that a variety of detection engines can be used in the agents, e.g., the IDIOT IDS [32]. *Transceivers* are entities that supervise all local agents, analyze their reports, aggregate the findings, and report them to one or more monitors. Moreover, transceivers have full control over the agents and can start, stop, and (re)configure them. *Monitors* can audit more than one transceiver. As monitors receive alerts from all over the network, they can perform data correlation over multiple hosts. However, the authors [11, 149] do not give further details on that. Monitors can also be organized hierarchically, so that lower-level monitors report to higher-level monitors. Finally, a central monitor on top of the hierarchy communicates with a user interface.

REQUIREMENTS AAFID utilizes a static hierarchical tree structure with a designated monitor taking over the root position and thus representing a SPoF. Hence, the system is not resilient against attacks or failures of these entities. Moreover, despite the fact that the authors consider low overhead and resilience against failures as important requirements, they do not address them in the implementation of their prototype. Finally, data dissemination as well as data correlation and aggregation are not addressed in AAFID.

EMERALD

The *Event Monitoring Enabling Responses to Anomalous Disturbances (EMERALD)* is another hierarchical CIDS proposed in [124]. It is designed for the monitoring of large enterprise networks and focuses on the detection of unauthorized access in domain resources.

OVERVIEW The system distinguishes three different layers: *service analysis*, *domain-wide analysis*, and *enterprise-wide analysis*. The service layer covers the detection of attacks across services and components within a single domain. The domain-wide analysis layer monitors multiple services and components. The enterprise-wide layer on top of the other layers, attempts to detect malicious activity across multiple domains.

Each of the aforementioned layers contains EMERALD monitors that use both signature-based and anomaly-based detection engines. Data dissemination in EMERALD is achieved via a subscription-based communication scheme. In more detail, each monitor may subscribe to others through a client/server-based asynchronous model and receive the respective alert data automatically. In addition, authors suggest that to ensure the security of the messages exchanged between monitors, a public key authentication may be used. Finally, there is

subsequent work on alert correlation techniques [166, 125], as seen in Section 4.1.3, that have been tested in an EMERALD environment.

REQUIREMENTS The hybrid detection engine used in EMERALD ensures a high accuracy for both known and unknown attacks. The system however does not provide any mechanism for the detection of insider attacks. Regarding interoperability, EMERALD provides an API that can be used to interconnect different monitoring tools. Nevertheless, this requires additional effort for the user of the system. In addition, no standardized data format for information exchange with other IDSs is used. Finally, according to the authors, ensuring reliable data delivery may increase the overall overhead of the subscription-based data dissemination mechanism.

HIDE

Hierarchical Intrusion Detection (HIDE) is another approach described in [192], that mainly focuses on applying novel anomaly detection techniques for the detection of malicious activity.

OVERVIEW *HIDE* arranges its monitors in a static hierarchical tier structure and employs anomaly detection via statistical preprocessing and neural network classification. Each tier in HIDE contains multiple monitors which are the so-called *Intrusion Detection Agents (IDAs)*. Each of them performs monitoring either on host or network level.

An IDA collects network traffic or host events and abstracts them to statistical variables and reports. These reports are then statistically checked and compared against the reference model maintained in the IDA. Afterwards, the result is taken and fed into neural network classifiers for further analysis and to determine if the traffic is normal or not. Finally, reports for higher tiers are generated and information is displayed via a local user interface.

REQUIREMENTS As HIDE uses anomaly detection techniques, higher false positive rates can be anticipated. In addition, the experimental results from Zhang et al. [192] indicate that low-volume attacks are hard to detect. Moreover, the authors mainly concentrate on describing the detection algorithm and on the selection of the best neural network. For this reason, many details are missing to properly assess HIDE according to the requirements from Section 3.2.

Summary

Decentralized CIDSs are intended for the protection of large networks by overcoming the scalability problems of centralized CIDSs. However, most of the observed decentralized CIDSs fulfill the proposed requirements only partially, as they usually contain one or more SPoFs. More-

over, decentralized CIDSs aggregate and correlate the data from lower levels and pass them over to the next level. At each level, the amount of data is reduced at the cost of lost information, which can result in a lower detection accuracy compared to a centralized approach. Most of the proposals in this category focus either on novel architectures (GrIDS, AAFID) or detection algorithms (HIDE). In these terms, EMERALD, although an early approach, is the most complete solution with respect the requirements proposed in this thesis. Lastly, Table 3 provides a summary of hierarchical CIDSs and their main building blocks (for an overall comparison of all surveyed CIDSs the reader can refer to Tables 5 and 6).

CIDS	*Local Monitoring*	*Correlation & Aggregation*	*Membership Management*	*Data Dissemination*	*Global Monitoring*
GrIDS	Host-based and Network-based	Monitor-to-Monitor	Hierarchical	Flooding	Malware Spreading
AAFID	Host-based	Monitor-to-Monitor	Hierarchical	?	Generic
EMERALD	Network-based, Hybrid	Monitor-to-Monitor, Filter	Hierarchical	Publish-Subscribe	Access Control
HIDE	Network-based, Anomaly	Single Monitor	Hierarchical, Static	Flooding	Generic

Table 3: Hierarchical CIDSs and their Building Blocks. A question mark ? symbol indicates unknown cases.

Distributed CIDSs

A distributed CIDSs architecture contains no central component nor hierarchy, as the tasks of the central analysis unit are distributed to all monitoring points. As a result, such a system that follows the P2P design principle, can scale with any number of monitors and thus can protect large networks. Moreover, the lack of a strict hierarchy, as in decentralized CIDSs, provides more freedom in interconnecting monitors and thus can be of benefit when encountering sophisticated and highly distributed attacks. However, depending on the specifics of the distributed CIDS, this can also be a drawback; as there is no hierarchy, there is no component in the CIDS that has a global view of the protected network.

As described earlier in this chapter, distributed CIDSs can be further classified with respect to the enforcement or not of a structure in the ID space of the corresponding P2P overlay. Hence, in the following we separate the analysis of distributed CIDSs accordingly.

Structured CIDSs

Structured CIDSs impose a structure on the participating monitors by organizing them using a DHT. A DHT provides guaranteed broadcast and search functionality when storing data in a distributed manner.

Most structured CIDSs that are given in the following make use of DHTs for the efficient storage of attack-related information, e.g., maintaining distributed blacklists. Others, e.g., INDRA, use a DHT for organizing their monitoring points.

However, this requires structure in terms of a fixed overlay neighborhood that is based upon the IDs of the CIDS monitors. Hence, this does not allow for flexible overlay connections. Moreover, since DHTs require that the data to be stored is mapped to the ID space of the DHT, any multi-dimensional data has to be reduced to a single dimension representation. Therefore, when storing IDS data in a DHT, it is necessary to select a single property as a key for the DHT. As a result, only single-attribute lookups and no complex multi-attribute searches are feasible. Moreover, as a consequence of the strict ordering of nodes and data in the ID space, most DHT algorithms and implementations cannot fulfill the domain awareness property. Due to this, privacy issues may arise when storing the monitored data.

INDRA The *Intrusion Detection and Rapid Action (INDRA)* is a P2P-based CIDS approach proposed by Janakiraman et al. [70]. The system does not focus on any particular type of attacks, but rather on generic malicious activity detection.

OVERVIEW INDRA organizes its monitors, so-called daemons, in a *Pastry*-DHT [136] and uses *Scribe* [23] as publish-subscribe mechanism for managing data sharing between daemons.

Monitors in INDRA act as both monitors and analysis units. When an attack is detected by an INDRA daemon, a proactive or reactive defensive action occurs. An INDRA daemon consists of four sub-components, *Watchers, Access Controllers, Listeners* and *Reporters*, that are described in the following:

- *Watchers* detect suspicious activities on a host or network level.
- *Access Controllers* are responsible for taking action against particular users, e.g., denying access to an account that is marked as compromised.
- *Listeners* aggregate the alerts generated by watchers and convey them to the access controllers.
- *Reporters* communicate with other hosts, in the sense of sending and receiving alerts to and from other hosts respectively.

The authors do not describe the detection mechanism that is used by *Watchers*. The information dissemination in INDRA is handled by *Scribe*. For every attack category, a *Scribe* group is created and nodes can subscribe to these groups. For instance, nodes may subscribe to the *Scribe* groups for *SSH* attacks and DoS attacks. In addition, the

authors claim that alternative models, such as rumor-spreading, can be used for the data dissemination, but without concrete suggestions on how to deploy them.

REQUIREMENTS INDRA tries to improve its accuracy by allowing the administrators to create plugins for new attacks. However, this manual intervention by the administrator needs significant effort as the aforementioned plugins have to be written manually (as code). Furthermore, a compromised monitor can reduce the accuracy of INDRA by producing fake alerts as a form of a DoS attack. For example, consider the case when a compromised peer claims that another peer within the trusted network is compromised. One of the main suggested defensive mechanisms is the creation of a blacklist. Hence, all peers in the network will insert a suspicious peer to their list, blocking it from any further communications.

LARSID Zhou et al. propose the Large Scale Intrusion Detection (LarSID) a P2P-based CIDS based on a publish-subscribe mechanism [195, 194].

OVERVIEW Every peer in the system is a monitor (via the usage of local IDSs) and also an analysis unit that creates a list with suspicious IP addresses and distributes it to the P2P network.

In order to be able to share alert information between different peers, the system employs a publish/subscribe mechanism on top of a DHT, i.e., a modified Pastry [136] DHT named *Bamboo* [132, 133]. The alert data in LarSID is in the form of lists of attackers' IP addresses. Each peer in the monitoring network is responsible for maintaining a watchlist for its local subnetwork, correlating subscription messages, and also generating notification messages regarding the identified malicious IP addresses. The system also utilizes a threshold policy. In more detail, if a certain number of monitors have flagged an IP address as malicious, then notifications are sent over the network. Otherwise, i.e., the number of detections of an IP is below the threshold, a new entry is created in the monitor's watchlist.

REQUIREMENTS As a distributed CIDS, the system scales to large networks. This is also supported by the experimental results in [194]. However, as reported in [193], certain nodes can become overloaded when a large number of attacks is originated from the same IP address. LarSID, assumes that all involved peers in the monitoring network are to be trusted. For this, it utilizes a Public Key Infrastructure (PKI) in order to ensure that participating nodes are authenticated. Moreover, communication between all monitors takes place over SSL. The main disadvantage of such an approach is its global

monitoring capabilities. LarSID can only detect attacks that involve a common source or destination IP address.

SIMILAR APPROACHES Many CIDS approaches have been proposed that are similar to the LarSID, e.g., *Komondor* [35], *Wormshield* [22], the P2P-based CIDS of Marcher et al. [104], and the Cyber Disease DHT (CDDHT) [88]. All of the aforementioned proposals are similar in terms of their structured CIDS architecture, although the underlying DHT implementation may differ. Moreover, they exhibit differences with respect to their specific purpose as well as in their key selection for the DHT. In more detail, their global detection varies from worm detection and containment (Wormshield), and DoS attacks, port scans, worms and Botnets (CDDHT) to more flexible ones (Komondor [35] or [104]). Finally, RepCIDN [62] is another DHT-based CIDS that mainly focuses on the construction of a reputation mechanism to handle internal attacks.

Unstructured CIDSs

Unstructured CIDSs provide more flexibility as no restrictions are imposed in selecting their overlay neighbors (peers to exchange data). In fact, this feature can be exploited in establishing flexible overlay relationships upon properties that are different from node IDs, e.g., based upon similarities in the monitored data. However, as there is no structured ID space, as with the case of structured CIDSs, data cannot be stored and retrieved efficiently as in structured CIDSs. For this reason, unstructured CIDS are not optimal for the distributed storing of attack-related information, e.g., blacklists, and to efficiently look them up again.

DOMINO The Distributed Overlay for Monitoring Internet Outbreaks (DOMINO) is described in [190].

OVERVIEW DOMINO utilizes a hybrid architecture with three kinds of entities: *axis overlay*, *satellite communities* and *terrestrial contributors*. Axis nodes act as both monitors and analysis units. Satellite communities and terrestrial contributors act as additional monitoring points that send their alert results to axis nodes for further analysis.

Axis nodes are the central component of the system and are connected via an (unspecified) overlay network. The axis nodes are assumed to be especially trustworthy and use a PKI for mutual authentication. Moreover, to counter insider attacks and fake alerts, each of them can enforce a threshold filtering upon received data. Satellite communities are smaller networks of satellite nodes that locally implement a version of the DOMINO protocol. They are organized in a hierarchy that is always led by an axis node. Finally, terrestrial contributors expand the system by adding non-trustworthy peers who

supply summaries of their detected attacks, without implementing the DOMINO protocol.

DOMINO utilizes signature-based IDSs, firewall rules (for intrusion response) and also honeypots for intrusion detection. Active-sinks are nodes that monitor a large number of unused IP addresses, with a low false positive rate that provides the possibility to produce signatures for unknown attacks. Finally, the alert messages are represented in XML format and are exchanged periodically.

REQUIREMENTS The hierarchical structure and the combined usage of both network-based IDSs and honeypots increase the overall accuracy of DOMINO. In addition, the alert messages are structured via XML, which ensures interoperability between different systems. Significant communication overhead may arise if the alerts' broadcasting period is shortened, compared to the one implemented, i.e., hourly alert broadcasting. Resilience against certain insider attacks is achieved by a static axis node architecture. However, this inflexibility in the axis overlay renders the system vulnerable to attacks on axis nodes. Moreover, multiple compromised and cooperating axis nodes can pose a threat to the overall system, especially as DOMINO contains no explicit countermeasure against insiders, e.g., a reputation system. Nevertheless, internal DoS attacks, e.g., by sending large amounts of alerts from a compromised axis node, can be mitigated by a threshold filtering mechanism at each axis node.

NEIGHBORHOOD-WATCHING Ramachandran et al. describe a P2P-based CIDS that uses mobile agents in a neighborhood-watch approach [130].

OVERVIEW Nodes in this system are arranged in an unstructured P2P network in which different kinds of agents are exchanged among peers to check for possible attacks.

Peers watch out for their neighboring peers and store critical information on each of them, e.g., checksums of critical data and operating system files as well as system binaries. If an anomaly is detected by a neighbor of an attacked peer, a voting process among all its neighbors takes place. When the majority agrees that intrusive behavior has been observed, all neighbors will attempt to protect themselves and at the same time notify and warn other peers in the network. To gather knowledge about neighbors in the P2P network, each peer sends different types of agents to its neighbors. These agents can perform a variety of tasks at the visited systems, e.g., establishing checksums of data files or looking for signatures of known viruses or worms. Moreover, agents are also used when voting about intrusive behavior in an overlay neighborhood.

REQUIREMENTS Besides a discussion of the system, the authors give no further evaluation of their proposal, nor any details on the methods used for the overlay establishment. At the same time, communication overhead issues may arise when a large number of agents is sent over the network to perform monitoring. The overall accuracy of the system could be low as information and data sharing is only done within a neighborhood level. Moreover, the authors do not provide enough insights for the detection mechanisms utilized. However, the voting process could have a positive effect in reducing the false positive ratio. Furthermore, the voting process can also provide resilience against insider attacks. Finally, sending agents to other systems to check crucial files that may conflict with the privacy requirement.

NETBIOTIC *NetBiotic* [182] is a distributed CIDS that is based on the *JXTA* P2P framework [64]. The focus of *NetBiotic* is not on detecting specific attacks, but rather on the fast creation of a network of interested peers for alert information exchange. Hence, the goal is to provide basic protection to participating peers, e.g., by detecting rapidly propagating malware. With respect to our terminology, each *NetBiotic* peer is both a monitor as well as an analysis unit and hosts a *notifier* and a *handler* component.

OVERVIEW At each peer, the *notifier* reads from log files written by security related applications, e.g., by a local IDS. On that basis, the notifier detects attacks, creates statistics of attacks, and finally transmits these statistics to other peers. In particular, the *notifier* calculates and transmits the percentage by which the average number of detected attacks differs from the average hits detected in earlier time intervals. If this percentage is significantly higher than a pre-configured threshold, the peer is considered to be under attack.

The *handler* is responsible for receiving messages from other peers and, if need be, to take action, like modifying the security settings of the end-user applications or to introduce new firewall rules. A significant difference from most of the other IDS proposals is that *NetBiotic* takes defensive actions only when the number of attacks detected is higher than the average, i.e., when an epidemic outbreak occurs. Moreover, one of the main features of *NetBiotic* is the ability to flexibly adapt the security policy, e.g., when the rate of past attacks is higher than the current ones.

REQUIREMENTS High detection accuracy rates are not the main focus of *NetBiotic*. As mentioned earlier, many attacks might remain undetected if there is no significant difference on the overall detection percentage. Interoperability is partially achieved as the system's architecture is supposed to be compatible with any IDSs that is able

to record its alerts in a log file. However, certain dependencies exist, such as the need for the incorporation of a parser to extract data from the log files. In addition, the countermeasures can be OS-specific. Finally, the decrease of the security level when there is a low attack detection rate appears interesting. However, this feature may be exploited by an insider to lower the overall security of a network and to subsequently attack it.

TRUST-AWARE CIDS Duma et al. proposed a trust-aware P2P-based overlay [41] collaborative intrusion detection based upon the JXTA framework [64]. This CIDS specifically addresses insider threats through the utilization of a trust-aware correlation engine and a dynamically adjustable trust management scheme. With respect to our terminology, each peer in this system is both a monitor and an analysis unit.

OVERVIEW The key component in each peer is the event manager that informs other peers for intrusion attempts and receives alerts from other peers. In addition, this unit provides filtering capabilities based on existing rules. The alert dissemination is done through targeted flooding to known peers. All exchanged alert messages are in the IDMEF format. To create trust among monitors, a list is maintained for acquaintance peers. Hence, every communication between monitors, e.g., the exchange of alerts, is evaluated and a score is generated for each peer. The higher the trust level of a peer, the bigger is its impact on others. The acquaintance list is dynamically updated and includes only the best available candidates that in addition had to pass a probation period. However, the trust mechanism requires that each peer is able to determine whether an intrusion event was genuine or a false positive. This cannot be always ensured and it highly depends on the accuracy of the employed local IDS. For the prototype implementation in [41], *Snort* has been used. As this is a signature-based IDS, the probability that a detected intrusion is actually an attack is high.

REQUIREMENTS The system is resilient to most of the insider threats, which includes Sybil and newcomer attacks. Nevertheless, some attacks are still feasible, e.g., sleeper attacks in which a highly trusted and long-term participating peer suddenly turns malicious. Such a peer would threaten the system as long as it remains in the acquaintance lists of other peers and can send fake alerts to others. The prototype system given in [41] makes use of the *Snort* IDS that cannot detect unknown attacks and that can be evaded as described in Section 3.3.1. However, the system is interoperable as Snort comes with IDMEF support for exchanging alerts.

WORMINATOR In [93, 94] Locasto et al. introduce the P2P-based IDS *Worminator*, whose peers act as monitors and each of them hosts a network-based IDS.

OVERVIEW *Worminator* exchanges compressed information via Bloom filters [19] with peers that are selected by a distributed correlation scheduling algorithm, called *Whirlpool*. Bloom filters are an efficient one-way data structure and are used to ensure privacy and compactness of the produced alerts. Upon a local alert, the corresponding information, e.g., source IP address and source ports, is inserted to a Bloom filter. Hence, the Boom filter represents a compressed list of suspicious hosts, a so-called *watchlist*. The watchlist is shared via the *Whirlpool* algorithm that creates dynamic neighborhood relationships in the overlay. Only neighbors exchange alert data via Bloom filters. However, the authors do not provide in-depth details on how this distributing scheduling algorithm works.

REQUIREMENTS Bloom filters are probabilistic data structures. False positive matches are possible especially with an increasing filling degree. However, there can be no false negatives, an element either has been included to the Bloom filter or not. Hence, when the number of included elements increases, also innocent hosts that have not been explicitly included to the Bloom filter could be identified as malicious. As a result, this affects the detection accuracy of *Worminator* beyond the actual detection mechanism. Nevertheless, the exchange of Bloom filters decreases the signaling overhead compared significantly to exchanging the uncompressed input data. Overhead is also reduced via the dynamic neighborhood formation in *Whirlpool*, at least in comparison to a full mesh distribution scheme or a random selection distribution scheme. As in many CIDSs, insider attacks are not covered by *Worminator*, therefore, malicious monitors can generate fake alerts and accuse other monitors to be malicious. Privacy of the sensitive data from the distributed alerts can be partially achieved by the utilization of Bloom filters, since data is compressed and hashed. In addition, the system uses a fixed list of participants for the alert correlation, so that only legitimate users are granted with Bloom filter access. However, an insider could gain access to the Bloom filter and be able to launch subsequent attacks. Such an attack could be to insert IP addresses of innocent hosts to accuse them for malicious behavior. Nevertheless, this can be avoided by protecting the Bloom filters via cryptographic means. Finally, an internal attacker could also query specific IPs to check if they had been detected in the Bloom filter.

QUICKSAND *Quicksand* is a CIDS that applies a distributed pattern detection mechanism for connecting distributed events manifested on a number of hosts [82].

OVERVIEW In a global monitoring level, *Quicksand* utilizes hybrid detection algorithms locally, and afterwards builds up signatures, so-called attack scenarios, that are used for connecting attacks that are distributed over the monitored network. To achieve this, each peer in *Quicksand* is a monitor that contains local IDSs with some pre-filtering capabilities and an analysis unit, the so-called *event correlation unit*. Correlation units are also responsible for intrusion response, e.g., reconfiguring firewalls upon the local detection of an attack. As *Quicksand* focuses more on the distributed pattern detection, it does not presume a specific detection engine. Monitors can locally apply signature-based as well as anomaly-based detection [83] techniques.

The system makes use of a centralized unit that stores new attack signatures and updates the signature databases of monitors. To represent signatures of distributed attacks, the authors introduce the Attack Specification Language (ASL) for describing subsequent intrusion steps as blocks of patterns. This allows to express complex relationships between distributed events on different hosts. Whenever an event on one host induces communication that leads to another event on another host, both events can be ordered and are set into relation in ASL. Afterwards, each attack scenario described in ASL is transformed to a directed and acyclic pattern graph. Nodes in the graph correspond to the distributed events that take place at different hosts. Connections and relationships are represented as edges, whereby one node constitutes a particular event and its successor node is the immediate successor of that event in the ASL. Once established, the centralized unit disseminates these pattern graphs to the monitors. Finally, at each monitor, the respective event correlation unit can check for possible intrusions. For that, it receives pre-filtered streams of events from the local IDSs and executes a distributed misuse detection algorithm that detects the occurrence of attack patterns on the basis of the pattern graph. In their prototype, the authors combine a *Snort*-like signature-based detection with an anomaly-based detection mechanism from [83]. To ensure compatibility, the system applies IDMEF [39] for exchanging information between detection and correlation units as well as in between different monitors.

REQUIREMENTS In terms of accuracy, while hybrid detection can be used locally by *Quicksand*, this is not the case for its global distributed pattern detection scheme. In this case, signatures (attack scenarios) have to be created, so false negatives, i.e., related attacks that failed to successfully be connected, exist. Moreover, the system allows only tree-shaped patterns to be created globally. However, this decision might not be always realistic, as it excludes other kind of patterns that might be more suitable. The system remains scalable as detection and correlation are performed locally first. Subsequently, only necessary information is exchanged between monitors, without

the need for involving a central party in the detection process. Finally, Quicksand uses IDMEF for improved interoperability and the authors explicitly provide information for the integration of third-party detection engines and IDSs respectively.

Summary

Distributed CIDSs were created to overcome the scalability limitations of centralized CIDSs approaches as well as the limitations of hierarchical approaches. In such an architecture, no entity has a global view of the network. Thus, the challenge is to provide accuracy rates close to that of a centralized CIDS, while protecting larger networks in which a centralized intrusion detection is no longer feasible.

On the one hand, structured distributed approaches provide efficient storing and lookup functionality. At this level, proposals such as LarSID offer a scalable CIDS solution with a fair, yet one dimensional, global detection block. On the other hand, unstructured systems have the ability to couple nodes on the fly, but with a less efficient way to store and retrieve alert data. In this direction, approaches such as DOMINO and the Trust-aware CIDS provide some promising and interesting properties [190, 41] Table 4 provides a summary of distributed CIDSs and their main building blocks (for an overall comparison of all surveyed CIDSs the reader can refer to Tables 5 and 6).

CIDS	*Local Monitoring*	*Correlation & Aggregation*	*Membership Management*	*Data Dissemination*	*Global Monitoring*
INDRA	Network-based, Signatures	?	Distributed, Structured	Publish-Subscribe	Generic
LarSID	?	Monitor-to-Monitor, Similarity	Distributed, Structured	Publish-Subscribe	Malware Spreading, DDoS
DOMINO	Network-Based and Honeypots	?	Distributed, Unstructured	?	Generic
Neighborhood Watching	Network-Based, Anomaly	?	Distributed, Unstructured	Selective Flooding	Generic
NetBiotic	Host-based	✗	Distributed, Unstructured	Selective Flooding	Malware Spreading
Trust-aware CIDS	Network-Based, Signatures	?	Distributed, Unstructured	Selective Flooding	Insider Attacks
Worminator	Network-Based, Hybrid	Single Monitor	Distributed, Unstructured	Selective Flooding	Malware Spreading
Quicksand	Network-Based, Hybrid	Monitor-to-Monitor	Distributed, Unstructured	Selective Flooding	Generic

Table 4: Distributed CIDSs and their Building Blocks. *The x marks ✗ indicate that the respective building block is not available, while a question mark ? symbol indicates unknown cases.*

Qualitative Comparison

In this section, we give an overall comparison of the surveyed solutions of CIDSs by focusing on the individual proposed requirements

given in Section 3.2. Table 5 provides an overview about all surveyed CIDSs from this chapter according to their employed building blocks (cf. Section 4.1). Table 6 provides a summary of our findings by listing all discussed CIDSs and comparing them to the requirements from Section 3.2.

CIDS	*Local Monitoring*	*Correlation & Aggregation*	*Membership Management*	*Data Dissemination*	*Global Monitoring*
DIDS	Host-based	Single Monitor	Centralized	Central Entity	Generic
SURFcert	Honeypots	✖	Centralized	Central Entity	Generic
CRIM	Network-based	Single Monitor	Centralized	Central Entity	Attack Prediction
DIDMA	Network-based, Signatures	Monitor-to-Monitor, Similarity	Centralized, Static	Selective Flooding	Generic
GrIDS	Host-based and Network-based	Monitor-to-Monitor	Hierarchical	Flooding	Malware Spreading
AAFID	Host-based	Monitor-to-Monitor	Hierarchical	?	Generic
EMERALD	Network-based, Hybrid	Monitor-to-Monitor, Filter	Hierarchical	Publish-Subscribe	Access Control
HIDE	Network-based, Anomaly	Single Monitor	Hierarchical, Static	Flooding	Generic
INDRA	Network-based, Signatures	?	Distributed, Structured	Publish-Subscribe	Generic
LarSID	?	Monitor-to-Monitor, Similarity	Distributed, Structured	Publish-Subscribe	Malware Spreading, DDoS
DOMINO	Network-based and Honeypots	?	Distributed, Unstructured	?	Generic
Neighborhood Watching	Network-based, Anomaly	?	Distributed, Unstructured	Selective Flooding	Generic
NetBiotic	Host-based	✖	Distributed, Unstructured	Selective Flooding	Malware Spreading
Trust-aware CIDS	Network-based, Signatures	?	Distributed, Unstructured	Selective Flooding	Insider Attacks
Worminator	Network-based, Hybrid	Single Monitor	Distributed, Unstructured	Selective Flooding	Malware Spreading
Quicksand	Network-based, Hybrid	Monitor-to-Monitor	Distributed, Unstructured	Selective Flooding	Generic

Table 5: CIDSs and their Building Blocks. *The x marks ✗ indicate that the respective building block is not available, while a question mark ? symbol indicates unknown cases.*

ACCURACY Foremost, the task of a CIDS is the detection of attacks, therefore maximizing the *accuracy* of this detection is the most important requirement for CIDSs. With respect to the proposed building blocks of a CIDS, accuracy is mainly influenced by the employed local detection mechanisms (local monitoring) and the detection mechanisms that operate on shared data (global monitoring). Moreover, essential for the global monitoring are the employed data correlation and aggregation mechanisms. These mechanisms pre-process and merge subsets of the data obtained from different monitors as a basis for the detection mechanisms operating on them. It is expected that the class of centralized CIDS provides a higher detection accuracy than a decentralized or distributed CIDS. Centralized CIDSs and their employed detection mechanisms can operate on the full data set, whereas decentralized CIDSs operate on aggregated data and dis-

	CIDS	Accuracy	Overhead	Scalability	Resilience	Privacy	Self- config.	Interoper.	Dom. Awar.
Centralized	DIDS	Ø	×	×	×	?	×	?	×
	SURFcert	Ø	×	×	×	×	×	×	×
	CRIM	Ø	✓	×	×	?	?	✓	×
	DIDMA	Ø	×	Ø	?	×	Ø	×	×
Decentralized	GrIDS	×	✓	Ø	×	✓	?	×	×
	AAFID	?	?	✓	×	?	?	?	×
	EMERALD	✓	×	✓	?	?	?	Ø	×
	HIDE	Ø	?	✓	?	?	?	?	×
Distributed	INDRA	Ø	?	✓	Ø	×	×	×	×
	LarSID	?	?	✓	×	×	✓	×	×
	DOMINO	✓	?	?	×	×	?	Ø	×
	Neighborhood-Watching	?	?	✓	✓	?	✓	×	×
	NetBiotic	×	?	✓	×	×	✓	×	×
	Trust-aware CIDS	Ø	?	✓	✓	?	✓	✓	×
	Worminator	?	✓	✓	?	Ø	✓	×	×
	Quicksand	Ø	Ø	✓	×	?	?	✓	×

Table 6: CIDSs and the proposed requirements. *Checkmarks ✓ indicate the fulfillment of the individual requirement, x marks ✗ their non-fulfillment, average symbols Ø their partial match and a question mark ? symbol indicates unknown cases.*

tributed CIDSs employ detection mechanisms that operate on subsets of the monitored data.

However, the level of achieved accuracy is determined by the specific detection mechanism that is employed by the respective CIDS. To make it worse, most of the surveyed CIDSs lack an evaluation of their accuracy in detecting attacks. A comparison of the surveyed CIDSs would even require to quantitatively evaluate all surveyed CIDSs in a comparable setting, which is simply not feasible. For this reason, it is difficult to state which approaches meet the accuracy requirements.

As centralized CIDSs operate on the full data set, we presume them to meet the accuracy requirements. Decentralized and distributed CIDSs are considered to have a lower accuracy and thus meet the requirements only partially. However, several of the discussed systems focus more on the architectural level than on the actual mechanisms for detecting attacks. For this reason, we consider these systems, e.g., NetBiotic and AAFID, to have a rather low accuracy and thus assumed not to meet the accuracy requirements. Among the decentralized approaches, we only assume EMERALD to have a sufficiently high accuracy as it utilizes a hybrid detection mechanism.

Among distributed CIDSs, we presume INDRA and NetBiotic to have a low accuracy in detecting attacks. INDRA makes use of simplistic detection methods and regarding NetBiotic, accuracy is not in the main scope of the system, but rather the creation of a network of peers for fast information exchange. The only distributed CIDS that we assume

to provide a good accuracy is DOMINO. This approach exhibits hybrid local monitoring via a combination of honeypots, dynamic firewall rules and network-based IDSs. However, in DOMINO this comes at the expense of a significant computational and communication overhead.

OVERHEAD The communication overhead created by a CIDS is a direct result of the employed data sharing and dissemination mechanisms. Among the discussed CIDSs, a multitude of techniques is used for this purpose, e.g., reverse multicast, publish-subscribe methods, and flooding mechanisms. However, all of these techniques have advantages and come with inherent drawbacks. For instance, while publish-subscribe algorithms seem to be promising for a deployment in CIDSs, it is not trivial to select the subscription criteria on which basis monitors subscribe to data. Flooding mechanisms come at high signaling overhead but cause monitored data to be available throughout a CIDS overlay and thus can result in an increased accuracy. There is a trade-off between the tolerated or caused signaling overhead and other requirements such as the resulting accuracy.

As can be seen in Table 6, we assume that most of the discussed CIDSs cannot meet the requirement of minimal overhead. However, to compare them with each other and to assess their overhead, similar to their accuracy, an extensive quantitative evaluation of all discussed systems would be required. As this is not possible, we can only base our assessment on the architecture of the observed systems, their design choices, and their limitations. For instance, the CIDS Worminator tries to minimize the communication overhead by the usage of Bloom filters and by exchanging them in between monitors via a certain scheduling algorithm. However, the data reduction by Bloom filters comes at the expense of a decreased accuracy.

SCALABILITY Scalability is a fundamental requirement, as CIDSs are intended to protect large networks. From the CIDS building blocks, the membership management is assumed to have the biggest influence on the scalability of CIDSs. Centralized approaches, as the name implies, utilize a centralized membership management and thus cannot scale to large networks. The central analysis unit represents a SPoF and a bottleneck. There are also several decentralized CIDSs that employ a centralized membership management, e.g., GrIDS, and thus do not scale as well. Scalable, decentralized systems with interesting architectures are the AAFID, that creates a hierarchical tree structure, and EMERALD, that divides its monitored space in a multi-layer fashion. The class of distributed systems is assumed per definition to be scalable. All distributed CIDSs summarized in this chapter seem to be free of a SPoF and bottlenecks. However, it is again difficult to assess the scalability of the discussed CIDSs individually, as most of them have not been described in sufficient depth.

RESILIENCE A CIDS has to be resilient to failures and attacks, which also includes insider attacks from compromised and malicious system components. Hence, when under attack, resilient CIDSs need to stay available or at least provide graceful degradation. By their nature, centralized CIDSs cannot be assumed to be resilient. An attack on their central component will bring down the system immediately. Decentralized CIDSs are more resilient against failure and attacks. However, a failure of the central analysis unit on top of their hierarchy can result in service unavailability. While there are mechanisms for automatic restoration of tree structures in case of failures, none of the surveyed systems makes use of them. Furthermore, in many cases, e.g., HIDE, the overall architecture appears to be static. Most of the observed distributed CIDSs utilize well studied P2P protocols for membership management, e.g., DHTs. These approaches are well studied and provide mechanisms for graceful degradation and fast restoration after failures. Hence, we assume them to be resilient to failures and external attacks.

However, most of the CIDSs discussed in this chapter are vulnerable from insider attacks. Only the *Trust-aware CIDS* [41] and the *Neighborhood-Watching* approach from Ramachandran et al. [130] provide countermeasures against malicious insiders. Hence, besides them we assume no other of the surveyed systems to be fully resilient and thus we rate them only as partially resilient.

SELF-CONFIGURATION Self-configuration is a pre-requisite for a resilient system as it allows for an automatic system restoration, e.g., after a failure or an attack has taken place. In addition, self-configuration avoids a manual and error-prone configuration of CIDSs. Distributed approaches inherently provide self-configuration mechanisms, e.g., via well-researched P2P mechanisms. DIDMA is the only centralized CIDS that, partially, offers self-configuration methods, e.g., agents are able to operate and self-configure themselves in the case of a failure of their dispatcher. For all other centralized CIDSs and also the decentralized CIDSs discussed in this chapter, it is not clear to which extent they support self-configuration.

PRIVACY Privacy in terms of not disclosing data to unauthorized sources is also an important pre-requisite when deploying a CIDS in larger scale and across different domains. However, only a few of the discussed systems include privacy-protecting mechanisms. For instance, *Worminator* partially achieves this requirement with the combination of utilizing Bloom filters along with a trusted list of participant peers. In addition, *GrIDS* exhibits a built-in privacy-protection mechanism due to the way it handles its hierarchical architecture. In more detail, each department is able to only observe activity that is restricted within its boundaries.

INTEROPERABILITY A CIDS should also be interoperable with other CIDS deployments. For that, several of the observed systems, e.g., the *Trust-aware CIDS* [41] or *CRIM* [34], make use of standardized formats for data exchange, e.g., IDMEF.

DOMAIN AWARENESS Finally, it is important for a CIDS to be able to take into account constrains that originate from the existence of security policies (e.g., the inability to disseminate alerts to all subnetworks). However, as it is depicted in Table 6, none of the surveyed CIDSs exhibits such domain awareness capabilities. This thesis contributes to the fulfillment of this requirement by proposing a novel CIDS in Chapter 9.

SUMMARY

To sum up, centralized CIDSs have the potential to offer the highest accuracy, but do not scale. Hence, they can only be used for the protection of small infrastructures, e.g., small corporate networks. For larger networks, such as a smart grid or large corporate infrastructures, more scalable solutions are required. Decentralized CIDSs seem to be suitable at first sight, but they are vulnerable from attacks and provide a lower accuracy than centralized systems. In terms of overhead, decentralized CIDSs' performance is highly dependable in regards to the utilized building blocks for correlation and aggregation. Most of the distributed IDSs are scalable to large networks and are resilient to attacks, but this comes at the expense of more overhead and an accuracy degradation compared to centralized systems. Hence, especially in this category, a lot of challenges remain to be addressed by future research.

However, especially in the class of distributed CIDSs, many interesting systems have been proposed so far. For instance, DOMINO utilizes a hybrid architecture, by combining a P2P-based core of especially trustworthy nodes with less trustworthy monitors that are organized in hierarchies. Nevertheless, as mentioned in the detailed analysis of DOMINO in Section 4.2.3.2, this might create inconsistencies with regard to our requirements, especially with respect to resilience against insider attacks. In addition, approaches from the structured CIDSs class, such as LarSID, demonstrate the efficiency and scalability of P2P CIDSs when the detection of certain attacks is required, e.g., DDoS.

The main finding of this chapter is that none of the observed systems can satisfy all requirements for CIDSs. While most distributed CIDSs are scalable and resilient, we assume that distributed approaches provide a lower detection accuracy than centralized or decentralized CIDS and induce too much overhead for a practical deployment. However, we also need to admit that our analysis of the surveyed CIDSs has been difficult, because almost none of them have been evaluated

in large-scale or in real-world environments. This clearly shows the need for more research in collaborative intrusion detection, especially in the area of distributed CIDSs.

Part II

ALERT DATA CREATION

The second part of the thesis presents contributions in the area of alert generation in the context of intrusion detection. First, Chapter 5 introduces *HosTaGe*, the first mobile honeypot. The honeypot is capable of emulating various protocols and systems, detect attacks, and also to generate respective signatures to be fed into IDSs. Chapter 6 discusses *TraCINg*, a cyber-incident monitor that makes use of *HosTaGe* sensors, and the lessons learned from a long period of deployment. Lastly, Chapter 7 proposes a toolkit for the automatic generation of synthetic, yet realistic, intrusion detection datasets.

5

HOSTAGE MOBILE HONEYPOT

In the previous chapters the thesis has argued for the necessity of innovative mechanisms for generating alert data. In this context, the current chapter proposes a novel honeypot for mobile devices, called *HosTaGe*, which is able to generate (alert) signatures on the fly. The remainder of this chapter is organized as follows: In Section 5.2, the design of the system is described along with justification behind the various design choices that were made. In addition, the section discusses aspects with regard to the different protocols emulated by the honeypot. Furthermore, Section 5.3 presents the evaluation of the system with a focus on its accuracy and applicability. Section 5.4 summarizes and concludes this chapter. Finally, Figure 9 depicts the overview of the chapter with regard to the overall thesis structure.

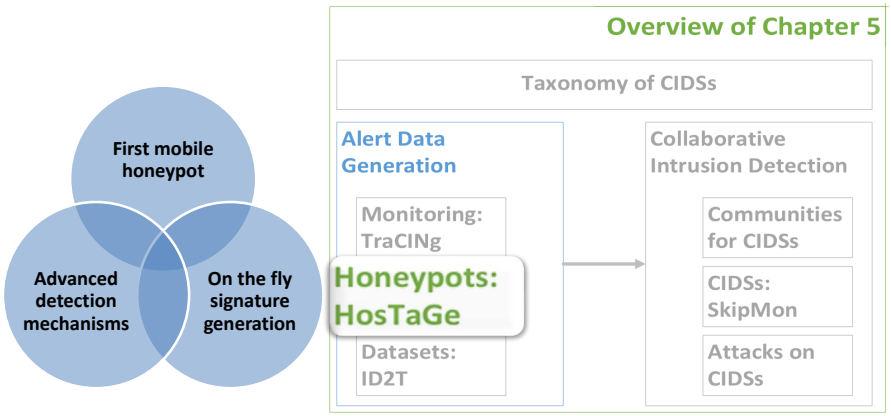


Figure 9: Overview of the Chapter and key contributions.

INTRODUCTION

The dependence of our society on IT networks is constantly increasing. In addition, the emergence and interconnectivity of Industrial Control Systems (ICSs) creates a plethora of security challenges that need to be addressed. For instance, recent highly sophisticated and tailored attacks against these systems, e.g., Stuxnet [86] and Flame [180], highlighted this fact.

Besides the traditional approach of deploying IDSs (see Chapter 2.1), honeypots can also assist to increase the overall detection accuracy of a monitored network. Honeypots (see Chapter 2.2) are systems whose only value is to be probed, attacked and compromised [151]. Their purpose is to attract malicious users, study their activities and, at the same time, reduce the attack surface. It is important to note that since honeypots do not feature any other purpose, by definition, any interaction with them is considered an attack. Thus, they exhibit a low false positive rate as all incoming traffic is considered malicious.

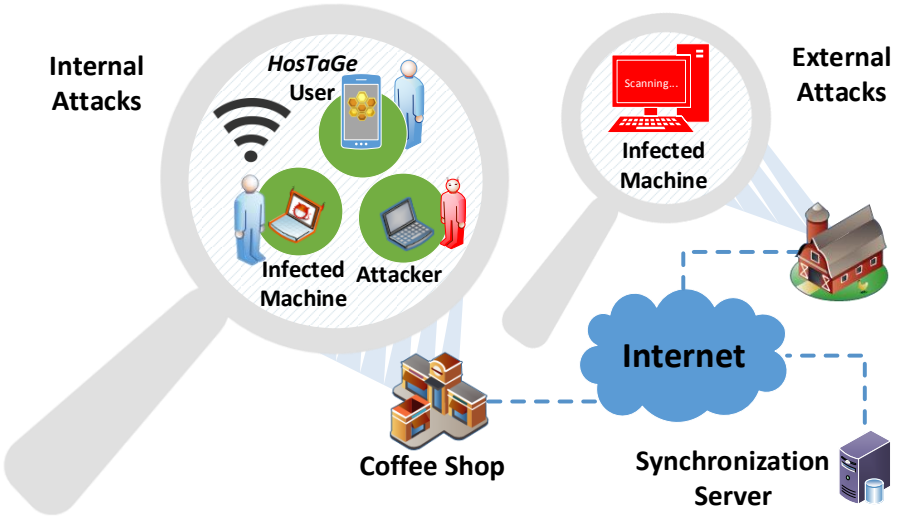


Figure 10: Attack surfaces and collaborative capabilities of HosTaGe.

This chapter, proposes *HosTaGe* a honeypot for mobile devices. The honeypot emulates all major protocols that are commonly used by malicious entities to perform attacks. The main contributions of this chapter are summarized in the following:

- Mobile Honeypot: *HosTaGe* (which stands for Honeypot-To-Go) is the first mobile honeypot. The introduced *mobility* allows for honeypots to travel and collect local threats before they become global. In addition, it serves as a ready-to-use security mechanism, for network administrators and security experts, to examine the security status of their network.

- Attack Surfaces: *HosTaGe* is intended for detecting attacks in a variety of situations. This is depicted in Figure 10; the honey-

pot can be utilized to detect both internal and external attacks. Internal attacks refer to malicious activity that originates from inside the monitored network (e.g., a malicious insider or an infected machine), while external attacks involve adversaries that come from outside the network (e.g., from the Internet).

- Alert Correlation: *HosTaGe* is able to identify attacks that originate from the same entity (i.e., unique IP address) and make use of multiple protocols. These attacks can realistically describe cases of Advanced Persistent Threats (APTs) and tailored real world attacks [148].
- Signature Generation: The honeypot is able to generate and extract signatures from detected attacks, which can subsequently be imported into other security mechanisms such as IDSs and anti-virus scanners.
- Collaboration: *HosTaGe* further takes advantage of its mobility by collaborating between different (*HosTaGe*) instances. Honeypots send their alert data to both a central server and to all other deployed honeypot instances.

Besides the aforementioned contributions, the honeypot was designed in a user-centric way to assist non-expert users. Moreover, *HosTaGe* offers some unique properties such as the ability to evade detection. In addition, it can emulate various Operating Systems (OSs) and dynamically change the OS with respect to the needs of the user. Lastly, the honeypot is one of the few that support protocols that are intended for the communication of ICSs.

SYSTEM OVERVIEW

This section discusses the design of the honeypot, by providing a description of the architecture of the system and its Graphical User Interface (GUI). Afterwards, a comprehensive discussion of the protocol emulation is given, accompanied by a formal model and a description of the detection mechanisms that the honeypot exhibits.

Architecture

Figure 11, depicts an overview of the architecture of *HosTaGe*. The honeypot is written in Java and supports the majority of recent Android OS versions. It runs on top of the Dalvik Virtual Machine environment [46] and consists of several modules that are closely interconnected, namely *HosTaGe Core*, *Logger*, *Port Binder* and the *GUI*.

When the honeypot is started, the *HosTaGe Core* runs in the background as a *Service* and activates the *Emulator* submodule. This submodule then emulates the selected protocols and listens to incoming

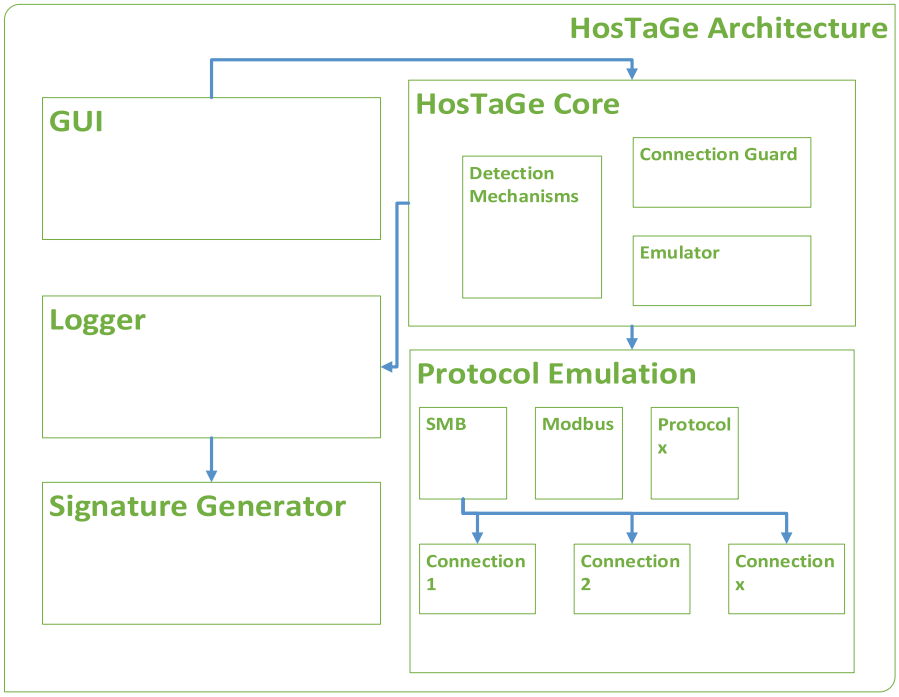
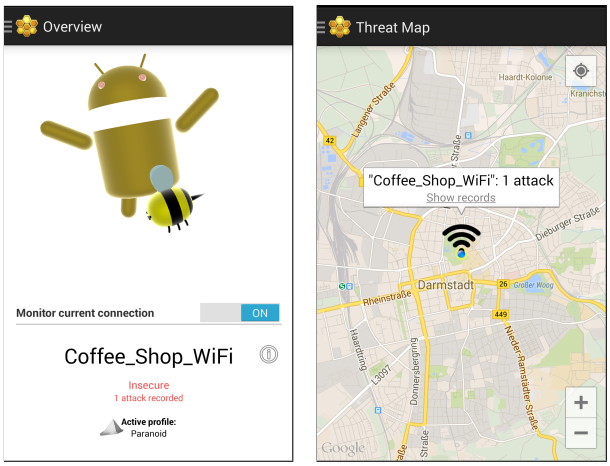


Figure 11: High level architectural view of HosTaGe.

connections in the respective ports that are associated to each of the protocols. In order to bind the sockets with the ports, the *Port Binder* which runs on the Linux Kernel level is called by *HosTaGe Core* with the port that it should listen on. Afterwards, the respective module listens for incoming connections. Upon receiving a connection request, *HosTaGe Core* alerts the *Emulator* submodule which in return calls the *Logger* to record all activities that are being observed. At the same time, the user is notified of the activities that are being detected via the *GUI*.

The core provides an interface for the activation or deactivation of the implemented protocols emulations. It also sends status reports to the GUI to provide the user with connection information. It consists of two submodules called *Emulator* and *Connection Guard*. As the name implies, the *emulator* is responsible for the simulation of protocols. It executes multiple threads (i.e., one thread for each selected protocol) that listen to incoming malicious connections to the respective ports. In addition, the emulator submodule activates the *Logger* module for logging all activities.

The *connection guard* prevents that the hosting device gets compromised. Furthermore, the *Connection Guard* is also responsible for blocking incoming connections when it suspects that the device is under attack, e.g., due to a DoS attack. In such a case it limits the maximum allowed incoming connections, from the same source IP and/or the same destination port. Besides that, established connections are also terminated after a certain time-interval.



(a) Threat Indicator (b) ThreatMap

Figure 12: Graphical User Interface of *HosTaGe*.

Lastly, the application, via the *logger* module, supports different formats for the generated log files. It also supports exporting the logs to a plain text file and/or to a database. In addition, for interoperability reasons *HosTaGe* can produce logs in the JSON[1] format for further processing of the alert data by other third-party applications.

With regard to the GUI and the *Protocol Emulation* a comprehensive description is given in the following.

Graphical User Interface

The GUI of *HosTaGe* has been designed to assist both advanced and ordinary users. In the following, some of the specific user-centric and user-friendly components of *HosTaGe* are described: *Threat Indicator*, *ThreatMap*, *Profile Manager*, *Alert Data Synchronization* as well as various other enhancements.

- *Threat Indicator:* The default view of the application is designed in such a way that *HosTaGe* immediately conveys the network security status via a *Threat Indicator* as seen in Figure 5.12(a). Via four different animations, the application indicates all possible states of the honeypot in the connected wireless network: 1) *Enabled* 2) *Disabled* 3) *Previously Attacked* 4) *Attacked*.

- *ThreatMap: ThreatMap* is a visualization mechanism for *HosTaGe* to geographically illustrate the recorded attacks via a multitude of techniques, e.g., GPS data and GeoIP discovery. The exchange of alert data results on the creation of a "heat map" of infected networks that the user can avoid as well as share with other

1 http://www.json.org

users (see the following *Alert Data Synchronization* paragraph). An example of the *ThreatMap* is shown on Figure 5.12(b).

- *Profile Manager:* The honeypot offers the user the ability to dynamically change the emulated OS. Through the *Profiles* menu, the user can easily choose from a list of profiles that offer combinations of OSs and services that the honeypot will emulate, e.g., a Windows XP machine, a nuclear power plant, etc. In addition, advanced options are also offered, e.g., the creation of custom profiles.

  Whenever a new OS or system is selected, *HosTaGe* stops the emulation of protocols that are no longer required and respectively begins the ones that are needed. Furthermore, the behavior of certain protocols might change to adapt to the needs of the emulated system. For instance, the implementation of the HTTP differs when emulating an Apache server from the case of emulating a nuclear power plant system.

- *Alert Data Dissemination:* As an additional proactive mechanism, the system allows alerts to be disseminated within the *HosTaGe* community. This collaborative feature (see Section 5.1) aligns with the concept of mobile honeypots; *HosTaGe* instances can learn local threats and disseminate the information before the threat propagates. This way, users are well-informed of (even) malicious wireless networks that have never been connected to in the past. Moreover, such alert information can be fed into the honeypot and visualized (e.g., via the threat map –see above).

  The dissemination of alert data can be performed in a device-to-device manner, e.g., via NFC and Bluetooth. Such an approach can be useful in the case where users or corporations want to privately exchange data. Furthermore, data can be exchanged in a broader way, as well, by making use of a centralized server (cf. Chapter 6). In both cases *HosTaGe* will ignore duplicate alert entries and will only accept unique, i.e., unseen, data.

- *Other Enhancements:* Many additional features exist in order to assist the user, such as *statistics* and alert *records* of the monitored networks. Moreover, to support advanced users and network administrators, additional settings are configurable. For instance, the user can activate specific ports and services through the *Services* menu. Finally, in-depth customization of the honeypot can be made via the *Advanced Settings* section within the *Settings* menu.

Protocols Emulation

The emulation of several protocols is of high importance for honeypots. This section discusses the various protocols that are emulated by *HosTaGe*. The protocols are logically classified into four classes, namely: *ICS*, *communication*, *monitoring* and *access* protocols.

ICS protocols

MODBUS PROTOCOL Among the various profiles supported by ICS devices, *Modbus* acts as a backbone for device communication. Modbus is a serial communications protocol initially published by Modicon for usage in its Programmable Logic Controllers (PLCs). It is now considered as the standard that connects industrial devices, and in particular it establishes communication between master and slave devices. As Modbus can also be used in a wireless mode for Remote Terminal Unit (RTU)-based communication, it is widely used in various ICS networks, e.g., nuclear power plants, gas and oil distribution, for communication between the PLCs.

Modbus has instruction sets for the interaction of devices. PLCs have *registers*[2] as memory units. The instruction sets are specified as *functions* which denote Read/Write (R/W) operations on the registers of the PLCs. Modeling the correct behavior of Modbus requires simulating the responses of the Modbus system for a command issued by a Modbus master device. *HosTaGe* simulates the Modbus communication protocols by implementing the request/reply mechanism. The Siemens PLCs also have registers as memory units, which store sensor data and application logic. The registers have the memory mapped to blocks at the register level. Through the Modbus protocol, the data in registers can be accessed and set. *HosTaGe* supports this request/reply paradigm for the memory mapped registers through dedicated data structures. Furthermore, the honeypot also supports the instruction set of Modbus, which is implemented as function codes. The instruction set involves registers' R/W, as well as the Modbus service diagnostics. Moreover, the Modbus implementation aims on avoiding detection from well-known reconnaissance tools. In more details, the Nmap[3] port scanning tool, with special scripts, is able to perform Modbus-specific detection and reconnaissance. Similarly, the Metasploit exploit toolkit provides also support for the detection and exploitation of Modbus. *HosTaGe* is also able to respond appropriately in all these cases. Section 5.3.3, discusses the importance of this in the context of honeypot evasion attacks.

2 For simplicity reasons coils are included in the term registers, even though strictly speaking they exhibit differences.

3 https://nmap.org

S7 PROTOCOL The *S7* protocol (S7 Communication) is a Siemens proprietary protocol utilized in PLCs of the Siemens S7-300/400 families. It is used for PLC programming, exchanging data between PLCs, accessing PLC data from SCADA systems and for diagnostic purposes. The protocol forms as a base for accessing the registers for R/W operations and also programming the PLC for user defined tasks. The S7 protocol has been implemented to simulate the communication with the PLC with the memory mapping of the Siemens S7 300.

Communication protocols

HTTP AND HTTPS PROTOCOLS *HTTP* is supported by the majority of the PLCs for remote configuration purposes. The HTTP web server in the PLC enables GET/POST messages for information exchange. This HTTP server is simulated by *HosTaGe* through its dynamic HTTP protocol implementation. A default welcome page that simulates the configuration website of a PLC is displayed when an adversary tries to access port 80. Similarly, *HosTaGe* supports the Hypertext Transfer Protocol Secure (HTTPS).

SMB PROTOCOL The *Server Message Block (SMB)* protocol enables network file sharing between devices of the same network. Previous analysis of the Stuxnet malware made evident that many attacks targeting ICSs make use of SMB to propagate. In Modbus, master systems control slaves and disseminate commands. These systems are usually control servers or host systems connected to PLCs or slaves that receive critical information and updates from the sensors placed on devices and PLCs. The master system is usually a Windows XP host connected in a LAN. By emulating the SMB protocol, along with Modbus, it is possible to detect not only external attacks but also malicious activities originating from the Intranet, e.g., the propagation of Stuxnet. Thus, in *HosTaGe* the SMB protocol has been implemented and customized to simulate a Modbus master system. The customization involves using the SMB protocol to simulate *HosTaGe* as a shared network drive to which a malware will try to propagate after infiltrating a host machine. The propagated file is detected, by the so-called *file injection* module, and its hash value is sent to the VirusTotal[4] sandbox for further analysis.

MYSQL A number of malware and adversaries attempt to attack MySQL servers. Such attacks are usually password guessing attacks or brute force attempts. The honeypot is able to detect such malicious behavior by emulating a basic MySQL server and replying to a number of basic commands.

4 https://www.virustotal.com

FTP PROTOCOL Similarly to the aforementioned MySQL case, *HosTaGe* enables support for the File Transfer Protocol (FTP) by emulating a respective FTP server.

SIP PROTOCOL Attacks on the Session Initiation Protocol (SIP) have been increasing over the last years [45, 117]. The honeypot can simulate the basic functionalities of a Voice Over IP (VOIP) server by making use of the *SIP* protocol.

Monitoring protocols

SNMP PROTOCOL The Simple Network Management Protocol (SNMP) is widely utilized for monitoring network devices for administrative purposes. For instance, the Siemens S7 family of PLCs supports the configuration of client devices through *SNMP*. This allows to remotely manage devices on the network. The SNMP protocol has been implemented (using the open source snmp4j library) to simulate an SNMP agent working on the Modbus slave profile and SNMP manager on the master profile.

SMTP PROTOCOL The Simple Mail Transfer Protocol (SMTP) is also widely used for various purposes. *HosTaGe* is emphasizing on the utilization of the protocol in the ICS context. For example, *SMTP* is used as a notification system for ICSs; it is utilized to notify devices about changes that trigger tasks. The SMTP service implemented in *HosTaGe* does not provide the notification service, rather, it provides a very basic protocol emulation for simple service discovery.

Access protocols

TELNET PROTOCOL The *Telnet* protocol allows accessing a basic shell on devices in order to read memory, delete files and execute commands. The Siemens S7 PLC also supports Telnet; users or applications can communicate with the PLC for file and backup operations. As such, *HosTaGe* supports this protocol by providing shell emulation for the attackers.

SSH PROTOCOL Secure Shell (SSH) is one of the most common access protocols. *HosTaGe* supports a basic simulation of SSH by emulating basic commands and offering the adversary a simple terminal interaction.

The remainder of this section goes deeper into the detection and signature generation mechanisms of *HosTaGe*, by first providing the fundamental formal model, in the form of Extended Finite State Machines (EFSMs), followed by a discussion of the signature generation.

Formal Model

The detection mechanisms in *HosTaGe* are formalized with the adaption of an Extended Finite State Machine (EFSM) model. An EFSM has all the properties of a normal Finite State Machine (FSM) with the added feature of utilizing arbitrary `if`-conditions, instead of only true or false conditions, to specify how a state transitions to a new state [50]. The formal model of the proposed detection mechanism is given by the *Attack Detection* EFSM $M = (S, s_0, I, O, V, P, \delta, \lambda)$, illustrated in Figure 13, where S is the set of all states, s_0 the initial state, I is the input, O is the output, V are the variables, P are the predicates, δ a set of transitions and λ the set of outputs generated by transitions.

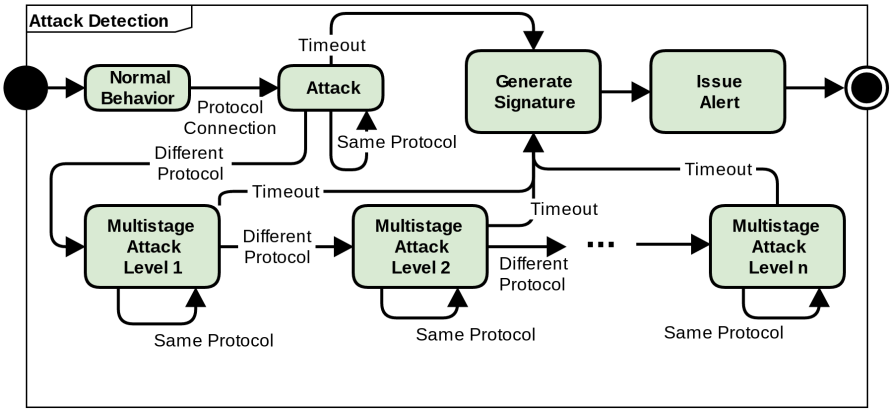


Figure 13: EFSM of the attack detection and signature generation mechanism.

The set of all states is represented with S. The EFSM starts in the *Normal Behavior* state, represented by s_0. If any protocol communication is detected by the honeypot, the EFSM transitions to the *Attack* state. For as long as the same protocol attack is observed, the state remains the same. If a timeout occurs the EFSM transitions to the *Generate Signature* state followed by the *Issue Alert* state. The signature generation is optional and will capture either a single attack or multistage attack types. After an initial attack, observing attacks originating from other protocols (but the same host) that have not yet been observed moves the state to the next *Multistage Attack Level* x, where x corresponds to the number of different protocols observed so far after the first one.

The inputs I, outputs O, variables V and predicates P are tightly linked together. State transitions are carried out whenever specific inputs $i \in I$ are received. These transitions may also generate an output $o \in O$. In the *Normal Behavior*, *Attack* and *Multistage Attack Level* x states, the Protocol Control Information (PCI) of the supported protocols are used as inputs and outputs. As such, {Modbus, S7, SNMP, HTTP, Telnet , SMB, SMTP, HTTPS, SSH, FTP} $\in I \in O$ for these states. The inputs I are not limited, however, to only ports. Distinctive PCI activities of interest on a protocol are also considered inputs. For in-

stance, the act of requesting a file through the SMB protocol is considered an input itself. V is a finite set of variables. These variables are used to construct a set of predicates P used for determining if a state transitions to another one. Each attack state holds a boolean variable $v \in V$ for each emulated port. If a particular port has been observed in the entire life of the EFSM, the corresponding variable for that port will be set to true. Besides variables, predicates P consist of the logic operator *AND* and the arithmetical operator $=$. We define the *Protocol Connection* predicate as the condition where a new protocol is observed without having observed other protocols yet. The *Different Protocol* predicate indicates, as the name suggests, that a new protocol has been observed after having seen at least one other. If any of these predicates is true a state transition takes place.

The final element of the model is the set of transitions $\delta(s_i, i, p) = s_j$ and the outputs $\lambda(s_i, i, p) = o$ generated by the transition itself. The set of transitions specify that whenever state $s_i \in S$ receives the input i and the predicate $p \in P$ is satisfied, the EFSM transitions to state s_j and outputs $o \in O$. The outputs are used by the *Generate Signature* state to create signatures for misuse analysis.

Detection Mechanisms in HosTaGe

The honeypot can distinguish between three different classes of attacks: *Single-Protocol Level Detection (SPLD)*, *Multi-Stage Level Detection (MSLD)* and *Payload Level Detection (PLD)*.

SPLD attacks refer to those that occur on a single protocol, e.g., HTTP connection attempts, without observing other protocols or any distinctive payload-level information. This is the simplest type of detection that can potentially still contain interesting results. For example, we can infer that a multitude of malware is trying to exploit the Telnet protocol (cf. Section 5.3) again due to the increasing amount of IoT devices such as IP Webcams. As this is the simplest of the three detection mechanisms, the description of its respective EFSM is omitted. In fact, the EFSM shown in Figure 13 can be seen as a generalized example of SPLD.

MSLD refers to attacks that originate from the same source and attempt to exploit different types of protocols within a small window of time. These type of attacks are identified by the honeypot with the EFSM shown in Figure 13, as described in Section 5.2.4. An important factor in MSLD is the time-window (*tw*) that determines whether an attack should be mapped as part of the SPLD or the MSLD class. This means that when the EFSM is on the *Attack* state and no further activity is detected (for a maximum of *tw*) a timeout will occur and the attack will be identified as SPLD. The *tw* can be adjusted with respect to the monitored network and its requirements. In this chapter (cf. Section 5.3) we experimented with the *tw* value of 15 minutes.

In practice, this suggests that when $tw = 15$, a transition from the *Attack* to the *Multistage Attack Level 1* state (via the *Different Protocol* predicate) is possible when a new attack occurs within the specified tw.

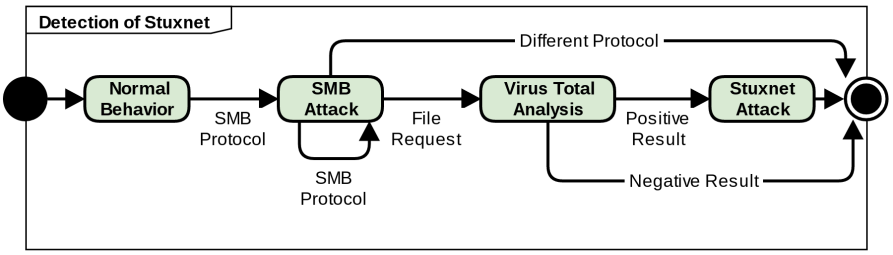


Figure 14: EFSM for PLD in the case of Stuxnet propagation.

The *Payload Level Detection (PLD)*[5] extends the applicability of the EFSM with respect to the inputs I. Referring back to the formal model, the outputs $o \in O$ from the *Attack* and *Multistage Attack Level* x states are used in the *Generate Signature* state to create signatures. Signatures are also EFSMs that comply with the presented model. As it was already mentioned, the input is not limited only to a port or protocol but also to potentially interesting/distinctive payload-level information.

Take Figure 14 as an example of an EFSM that represents a signatures generated by the PLD. This signature identifies Stuxnet attacks from the set of outputs O obtained from the *Attack Detection* EFSM shown in Figure 13. The *Detection of Stuxnet* EFSM assumes an initial *Normal Behavior* state and transitions to *SMB Attack* if an SMB protocol is observed. Stuxnet tries to inject an infected file through SMB. After a file is received, the file itself (or its hash value) is sent to VirusTotal and, if the file is indeed malicious, the EFSM transitions to the *Stuxnet Attack* state where the presence of Stuxnet is reported.

Signature Generation

As discussed in the previous section, upon detecting an intrusion, *HosTaGe* is able to generate signatures that can be utilized by IDSs. IDSs usually inspect all incoming packets for malicious content and make use of signatures to determine whether traffic is malicious or not. *HosTaGe* captures not only the attack packets, but also all the received connection requests at the protocol level. In addition, it records the entire connection tear-down that takes place between an adversary and the honeypot. This is important for both the signature generation process as well as the post-attack analysis that the user might want to perform.

HosTaGe utilizes specific modules to handle the aforementioned features (mainly the modules included in the logger, the signature gener-

5 *HosTaGe* currently supports PLD for the SMB, HTTP and Modbus protocols.

ator and the HosTaGe core – cf. Figure 11). The current version of the honeypot supports the signature generation process for the Bro IDS [120]. Bro is considered the state of the art in IDSs (see Chapter 2.1); it is highly adjustable and it is widely utilized especially for research purposes [95, 43].

The Bro IDS uses the so-called Bro language, an event-driven scripting language that can be used for extending and customizing the IDS's functionality. The Bro IDS signatures rely on packet data to check for the content to be matched for incoming packets. The *signature generation* module in *HosTaGe* is used to generate the signature files[6] for Bro. By specifying the protocols for the signature generation mechanism, the packet information of the connection is derived through the *attack records* module from *HosTaGe* to a template signature file which is used to generate signature files for Bro. The *attack record* module of *HosTaGe* generates signature files for a protocol.

An example of a signature generated by *HosTaGe*, for the Modbus protocol, is shown in the Listing 1. This signature, automatically created from *HosTaGe* and written in the Bro language, is able to detect the well-known Metasploit script for Modbus services identification (also see Section 5.3.3).

Listing 1: Modbus attack signature generated by *HosTaGe*.

```
signature modbus-signature{
    ip-proto == tcp
    dst-port == 502
    payload /\x21\x00\x00\x00\x00\x06\x01\x04\x00\x01\x00\x00/
    event "Modbus attack"
    }
```

EVALUATION

This section presents various experiments that demonstrate the detection capabilities of *HosTaGe* as well as its ability to generate signatures from the identified attacks. In addition, observed multi-stage attacks are discussed and some insights are given regarding the detectability of *HosTaGe*. The reader can also refer to Appendix A for some additional evaluation results of the honeypot with regard to its battery consumption and its ability to detect malware.

Honeypot Comparison

As a first evaluation step, *HosTaGe* is compared to the Conpot honeypot[7]. Both systems were deployed in a controlled environment with

6 With respect to the in the Bro IDS terminology *HosTaGe* can also generate security policy files.

7 The exact version of Conpot was "0.2.2".

no firewalls in between the honeypots and the Internet. The honeypots had similar IP addresses in the sense that they were in the same /24 subnet. The evaluation period for the tests was 12 weeks starting September 2015.

The results of the analysis are given in Figure 15. The results gathered from the two honeypots are compared for the HTTP, Modbus and S7 protocols. In addition, the section presents the results gathered for Telnet (even though Conpot does not support it) as Telnet is considered an important attack factor for ICSs networks [109]. The reader should note that the S7 protocol comparison presented in the figure is for a shorter period of time (i.e., 8 weeks) as S7's implementation in *HosTaGe* was completed in a later stage. As one can observe from the results, *HosTaGe* exhibits good detection accuracy in comparison to Conpot. In fact, in most cases *HosTaGe* detected more attacks than Conpot (e.g., HTTP, S7), while for the case of Modbus the number of detected attacks is almost equivalent.

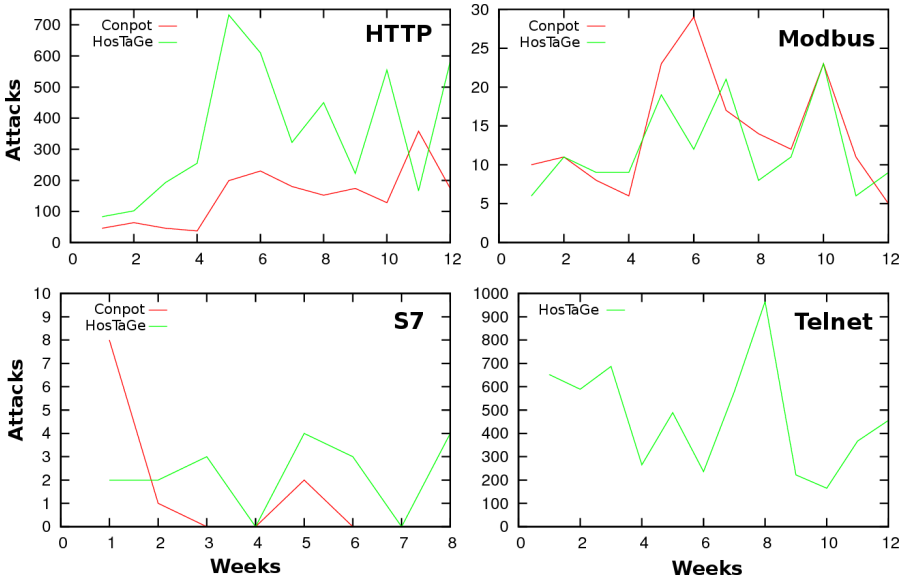


Figure 15: Comparison of detected attacks on *HosTaGe* and Conpot for HTTP, Modbus, S7 and Telnet. Note, that Conpot does not support the Telnet protocol.

It should also be noted that the purpose of this experiment was to compare the two honeypots and to identify automated attacks that target ICS networks. For this reason no advertisement of the honeypots was made in any way. However, as it is shown below, both honeypots were probed by the well known search engine Shodan.

Lastly, Figure 16 depicts the malicious unique IP addresses (and also their intersection) that were detected for each honeypot (for the HTTP and Modbus protocols). At a glance, the histogram shows that *HosTaGe* was able to consistently detect more unique IP addresses (of attackers) than Conpot.

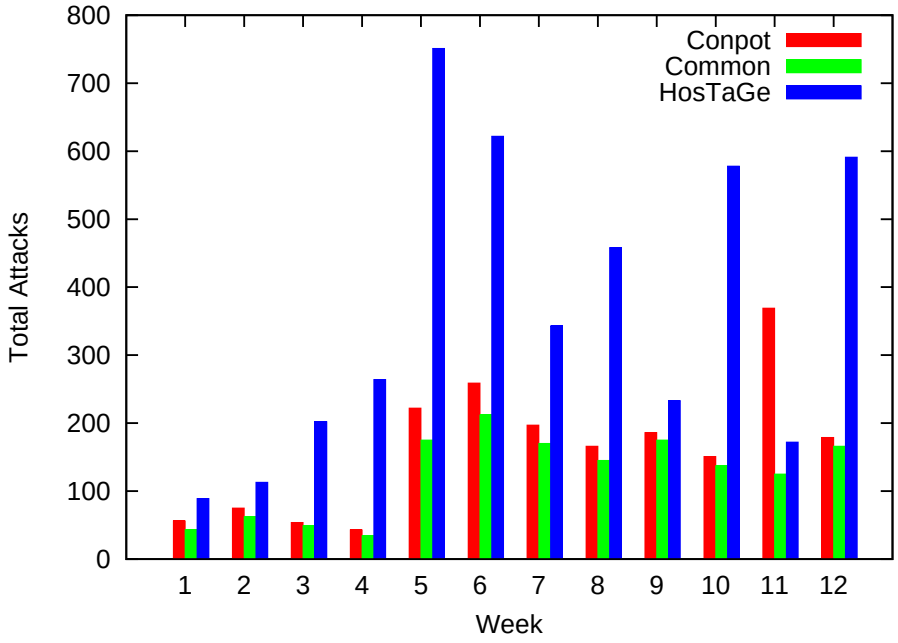


Figure 16: Comparison of unique and common malicious IP addresses targeting *HosTaGe* and Conpot

Multi-Stage attacks

During the observed period, *HosTaGe* also automatically detected three multi-stage ICS-specific attacks (within a time-window (*tw*) of 15 minutes). First, an attack from Iran was detected, which based on the Geo-IP information, appeared to be close to where Stuxnet was initially detected, i.e., the Tehran Nuclear Research Center. The attack included a portscan and sent an HTTP GET request to the honeypot. Subsequently, the honeypot detected an Nmap script that was trying to validate whether a Modbus service is indeed running on the host. The second multi-stage attack was originated from the Zhejiang Nuclear Industry in China. It started with a portscan, continued with an HTTP attack and finished with Modbus protocol requests that queried specific ICS services. The third interesting attack was from an area close to a power plant in Colombia. The attack stages consisted of a web-connection through HTTP, a general purpose scan of the network and finally an attempted Telnet connection to the honeypot.

Further experiments are required (e.g., for a longer observation period) to conclude on the aforementioned multi-stage attack results. Nevertheless, the fact that all the detected attacks appear to be originating from power plant locations suggests that the attacks could be triggered by existing (ICS-specific) malware that attempt to propagate further.

Honeypot Evasion

One essential requirement for honeypots is their ability to remain undetected; hiding the fact that they are not a real system but a honeypot. This section discusses how, in the aforementioned experimental setup, *HosTaGe* managed to remain undetected (while Conpot did not for example) from the Shodan[8] search engine.

Shodan is a specialized search engine that crawls the Internet and attempts to identify connected devices, e.g., IP web-cameras, ICSs, etc. [15]. Through the continuous process of crawling and indexing, the system creates an up-to-date database of systems and services exposed on the Internet. Recently, Shodan implemented detection mechanisms to identify honeypots. It performs a series of probes, and checks, and subsequently creates a score for each probed device. Based on this score value, Shodan determines whether a specific system is likely to be a honeypot or not.

A few weeks after the initial experiments, both *HosTaGe* and Conpot received probes from IPs that belong to Shodan. The protocols being probed were S7, Modbus, SSH and HTTP, among others. For the SSH and HTTP protocols the messages and requests were of low complexity; the HTTP protocol involved a GET request and the SSH an attempt to establish a connection. The S7 and Modbus attacks were more complicated. The conducted analysis indicates that the probes are, most likely, the results of modified Nmap/Metasploit scripts for ICS identification. These probes were able to identify Conpot as a honeypot but not *HosTaGe*.

In more details, the S7 attack involved queries regarding the device type, location, serial number, plant identification and module name. The Modbus attack involved fetching details of certain units (i.e., unit number 0 and 255) and their slave data. On the one hand, Conpot could not respond reasonably in all the aforementioned requests (either due to utilized static serial numbers or certain Modbus protocol simulation errors) and thus was classified as a honeypot. On the other hand, *HosTaGe* managed to respond successfully and hence remain undetected.

Signature Generation

The last part of the evaluation focuses on the signature generation mechanism and its effectiveness. That is, the applicability of the generated signatures is being examined.

The first goal is to generate multi-stage attack signatures. To achieve this, a *HosTaGe* instance is manually attacked in a controlled environment. Such attacks can be easily injected by contacting the honeypot via different protocols in a certain time window (see Section 5.2.5).

8 https://www.shodan.io

Attacks are additionally captured by the network packet capture tool Wireshark and saved for future reference (see below). Simultaneously, *HosTaGe* detects attacks and constructs signatures from both the protocol and payload-level interaction (cf. Section 5.2.5.1).

For determining the applicability of the generated signatures, two different tests are performed. First, the system is examined for false positives and afterwards for true positives. All tests utilize publicly available datasets of network traffic (in the form of pcap files). They consist of synthetic and real network captures[9], malware focused traffic[10], and honeypot captured traffic[11]

The first part of the evaluation is intended to determine whether the generated signatures introduce false positives. For this, the signatures are imported into the Bro IDS and the network traffic of the (unmodified) test datasets is replayed. With respect to the definitions in Section 5.2.5, a time-window of $tw = 15$ (minutes) is utilized for all the experiments. As it was expected, the analysis showed no signs of multi-stage attacks (false positives) detected by Bro.

Afterwards, each dataset is merged with the network traffic captured by Wireshark in the initial step (that includes the injected multi-stage attacks). Subsequently, each modified network file is again replayed while the Bro IDS is monitoring. In all cases, Bro successfully detected all of the injected attacks without generating any false positives.

SUMMARY

This chapter introduced *HosTaGe*, a mobile honeypot. The proposed system contributes in the areas of intrusion detection and alert generation in several ways.

HosTaGe is the first, low-interaction (see Chapter 2.2), honeypot intended for mobile devices. In addition, the concept of honeypots-to-go was introduced to support network administrators and regular users to assess the security status of their networks. The honeypot is also designed in a user-centric manner making it usable to a broad audience, deviating from related work which targets only advanced users.

Moreover, the honeypot provides the ability to emulate sophisticated ICSs and detect targeted malware, e.g., Stuxnet. Furthermore, *HosTaGe* formalizes the problem of attack detection and offers the ability to also detect multi-stage attacks by correlating attacks from the same source. To the best of the author's knowledge, *HosTaGe* is the first honeypot with such advanced correlation and detection capabilities (see Section 5.2.5).

9 Small and Big flows datasets: http://tcpreplay.appneta.com/wiki/captures.html
10 CTU-13 Dataset: https://stratosphereips.org/category/dataset.html
11 HoneyBot Dataset: http://www.netresec.com/?page=PcapFiles

This chapter also touches the topic of honeypot evasion; the evaluation results show that honeypots must be designed carefully to avoid detection. Lastly, the honeypot demonstrates advanced collaboration capabilities by being able to distribute its alerts to either other *HosTaGe* instances or to a central cyber-incident monitor (see Chapter 6).

6

TRACING CYBER INCIDENT MONITOR

The previous chapter introduced a security mechanism for detecting attacks and generating alert data. This chapter will complement and provide additional support for the aforementioned contribution while also contributing on alert data generation field in general. In particular, this chapter introduces and discusses *TraCINg*, a cyber incident monitor. The purpose of this contribution is to first create a platform for alert data generation and subsequently analyze the produced data with a focus on correlated attacks. The remainder of this chapter is organized as follows. Section 6.2 provides a description of the system's architecture and a justification behind the various design choices that were made. Section 6.3, gives insights and an analysis of the findings for a real-world deployment period of five months. Section 6.4 summarizes and concludes this chapter. Finally, Figure 17 depicts the overview of the chapter with regard to the overall thesis structure.

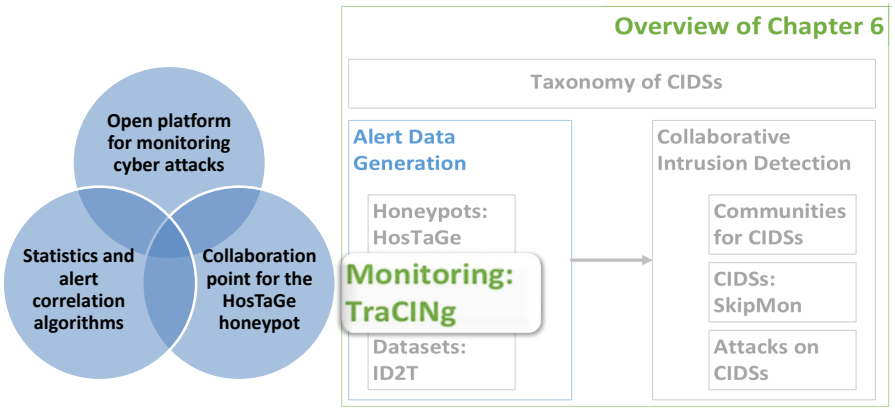


Figure 17: Overview of the Chapter and key contributions.

INTRODUCTION

With the increase of cyber-attacks in both numbers and sophistication, it is becoming evident that advanced techniques and mechanisms are required to identify new trends and patterns in the adversaries' strategies. This also implies that relying only upon traditional lines of defense, such as IDSs and dynamic firewalls alone, would not be able to provide a holistic coverage on detecting such novel and emerging patterns of attacks [172].

This chapter approaches the aforementioned challenges by proposing and discussing the deployment of a cyber-incident monitor. Cyber incident monitors are platforms utilized for supporting the tasks of network administrators and for providing an initial step towards coping with large amounts of alert data. Furthermore, such systems can have a rather diverse input flow, including, e.g., IDSs and honeypots.

As mentioned in Chapter 2, honeypots are systems whose value lies solely in being probed, attacked or compromised [151]. They can complement other detection mechanisms, e.g., IDSs, by providing a more active and in-depth view on attackers' activities. At the same time, by definition, honeypots produce zero false positives and they are able to detect even unknown and novel attacks.

Feeding honeypot alert data into a cyber-incident monitor provides a number of significant benefits. As mentioned above, in contrast to IDSs, honeypots do not produce false positives. Therefore, the security administrator can focus on real attacks rather than speculating from the observed results. In addition, detecting epidemic behavior, e.g., malware spreading, becomes more likely when analyzing attacks from several collaborating honeypots.

In essence, the inclusion of different types of alerts, i.e., from IDSs and honeypots, increases the affluence of new data to be processed by the administrators. However, due to the different focus and content of alerts generated by different systems, integrating these alerts into a single platform, creates challenges. Cyber incident monitoring systems are developed to overcome this problem by providing an aggregation and visualization interface to unify the various alert generators into a single system that assists the user in making informed decisions. Starting from a broad overview, users can decide to dive into specific alerts to assess a particular reported cyber incident.

This chapter introduces *TraCINg*[1], which stands for TU Darmstadt Cyber Incident moNitor, an open-source centralized cyber incident monitor that supports a number of different types of sensors. *TraCINg* is able to visualize attacks, provide statistics, present the geolocation information of malicious users, and perform several other user-centric operations. The design rational of the system is explained

1 https://www.tk.informatik.tu-darmstadt.de/de/research/secure-smart-infrastructures/tracing-tud-cyber-incident-monitor

along with a discussion of initial results from a five month real-world deployment period. Several interesting findings are shown in the results. First, many attack trends are identified, such as popular protocols and ports, the most persistent countries of origin, etc. Second, as it will become evident in Section 6.3 even a relatively small deployment of sensors, is adequate for detecting attacks manifested by the same source on multiple sensors. This correlation between attacks further motivates the need for collaboration between different monitoring points. In particular, the detection of targeted and distributed attacks can be significantly improved by applying correlation and aggregation algorithms on collaboratively generated sets of alert data.

ARCHITECTURE OF TRACING

In this section, a description of the architecture of *TraCINg* is provided, which includes four main parts: the *TraCINg core*, the *sensors*, the *GUI* and the *alert* output. Figure 18 gives a high level overview of the architecture of the system.

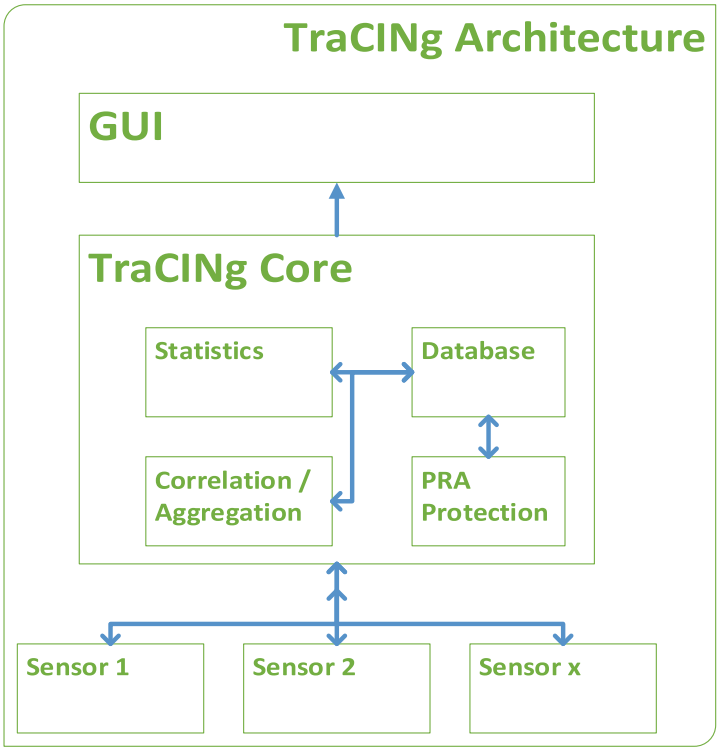


Figure 18: High level architectural view of TraCINg.

TraCINg Core

TraCINg is an open-source centralized cyber incident monitor that obtains alert data from geographically distributed honeypot sensors.

The current proof of concept of the system exhibits sensors deployed in three different continents, namely: Europe, Asia and the Americas. *TraCINg* follows a classic client-server architecture and uses an HTTPS server for receiving data securely. In addition, a Public Key Infrastructure (PKI) allows for the authentication of the sensors so that only certified sensors can send data to the system.

DATABASE All the received alert data is saved in a database that allows for further processing and analysis. For these purposes MongoDB was chosen as besides various interesting properties (e.g., support a high "write" load, location-based support for spatial data) it also supports dynamic querying and provides reasonable scaling [24].

STATISTICS, CORRELATION AND AGGREGATION The statistics along with the correlation and aggregation modules are responsible for analyzing the alert data. This varies from simple statistical functions, e.g., creating tables of "top" attacked ports and protocols, identifying the most common countries that are utilized by adversaries, to more sophisticated correlation algorithms. For the latter, the reader may refer to Section 6.3.3.

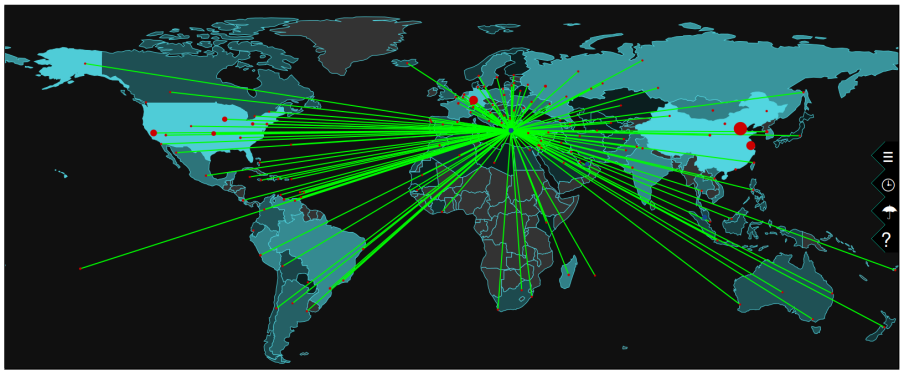
PRA PROTECTION Lastly, the Probe Response Attack (PRA) protection module of *TraCINg* applies novel techniques for the detection and mitigation of a specific type of attack that can result in the identification of the (network) location of the sensors. The reader can refer to Chapter 10 for a detailed description of the deployed PRA detection and prevention algorithms.

GUI

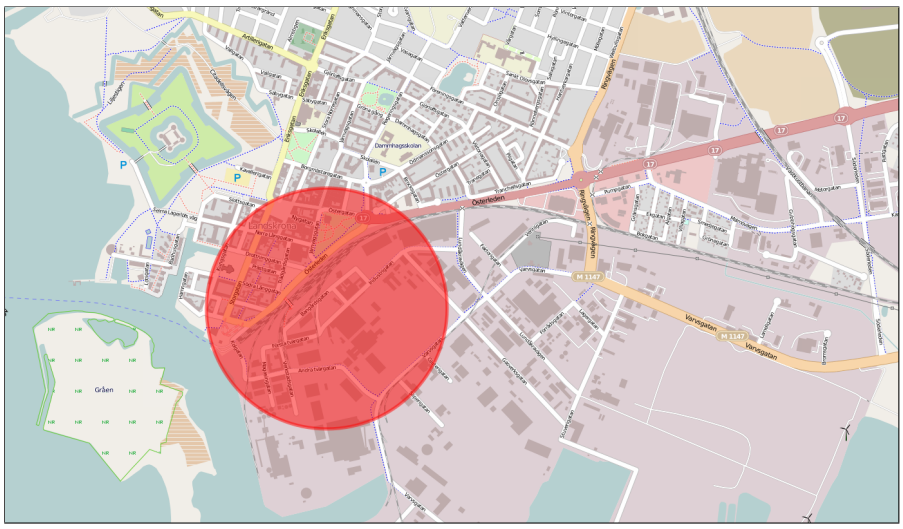
All the gathered alert information is available to the security administrators and users via a user-centric GUI with different types of views (two of them depicted in Figure 19) and functions, e.g., the visualization of statistics, on-demand detailed information, live and replay mode of the alert data, etc. The front-end, written in *node.js* [163], supports the integration of, e.g., OpenStreetMap [66] for the visualization of attack origins and targets (cf. Figure 6.19(b)) and VirusTotal [164] for obtaining additional information on identified malware. This latter property of supporting checks via the VirusTotal is particularly useful for quickly examining malware that were detected by the sensors.

Sensors

TraCINg supports and utilizes sensors that can be distinguished into two main types: *stationary* honeypots and mobile honeypots. Here, a



(a) The default TraCINg view.



(b) OpenStreetMap integration in TraCINg.

Figure 19: GUI examples of TraCINg.

short discussion of these types of sensors is given along with some implementation-specific details.

Stationery Honeypots

In contrast to mobile honeypots, stationery systems refer to honeypots that are static. As this is the most common type of honeypots *TraCINg* allows for their support with the only requirement being a certain alert formatting (see Section 6.2.4).

The current proof of concept of the system utilizes *dionaea*[2], a widespread low-interaction honeypot [147]. Dionaea honeypots are used in two different ways. First, honeypot instances are deployed on regular machines, i.e., Virtual Machines (VMs), as well as Raspberry Pis. This allows for an easy copying of systems, to be able to reuse them, and introduces convenient plug-n-play support for the sensors.

2 http://dionaea.carnivore.it

Second, dionaea sensors can be also deployed as cloud instances. This is important for being able to monitor attack traffic that may be specific to geographical regions. Thus, via the utilization of cloud providers, it was possible to deploy our sensors in different continents. Furthermore, cloud services provide resilience and uptime reliability for our sensors which ensures uninterrupted monitoring.

Mobile Honeypots

TraCINg also provides support for obtaining data from mobile honeypots, and specifically from *HosTaGe* as described in Chapter 5. The reader can refer to the aforementioned chapter for more details. At a glance, *HosTaGe* is a honeypot for mobile devices for detecting malicious wireless network environments. The driving idea behind *HosTaGe* is that mobile honeypots can detect local threats before they become global; subsequently, this provides the possibility of respectively mitigating such threats.

In addition, the combination of the collaboration techniques of *HosTaGe* and the alert synchronization with *TraCINg*, creates additional benefits for the users. First, one user can learn from others about the global security status of wireless networks in their city, or country. Moreover, the diverse data that is sent to *TraCINg* can also provide with valuable and more accurate input for the detection of correlated attacks and latest trends from the behavior of malicious users and malware.

Alerts

TraCINg supports input from any type of honeypot or IDS, with the only restriction being to utilize the respective input interface for the submitted alerts. Sensors need to convert their alerts into a defined JSON[3] format that is being supported by the input interface. Therefore, it is straightforward to add support for other honeypots. For instance, to enable *TraCINg* support for the dionaea honeypot, the respective alert export functionality was implemented as an internal dionaea module.

Another important aspect is the frequency of the exchanged alerts. In the current version of the system, alert data is sent instantaneously upon the detection of an attack by the honeypot to *TraCINg*, except for *HosTaGe* honeypots in which, the exchange frequency is determined by the user or by the respective auto-upload functionality. Take note that the frequency of the exchanged alerts could also lead to some attacks on the sensors (cf. Section 10 — *Probe-Response attacks*).

The generated alerts that are submitted to *TraCINg* contain the following attributes:

- *Time-stamp*: The date and time specifics of an attack.

3 http://www.json.org

- *Id*: A unique identifier for each alert.
- *Sensor Type*: A field that differentiates between different honeypots, e.g., dionaea, HosTaGe, etc.
- *IP*: The source and destination IP address of attacks (this information is excluded from the perspective of the end-users, i.e., from the GUI, to maintain privacy).
- *Ports*: The source and destination port of attacks.
- *Attack type*: The type of the detected attack, e.g., a portscan, a shellcode injection, etc.
- *Geo-location Information*: The geo-IP information, and the name of the city and country.
- *Authorization Status*: A boolean id that indicates whether a sensor possesses a signed certificate from the main TraCINg Certificate Authority (CA) via the usage of a PKI.
- *MD5 hash*: The MD5 hash of the malware collected from a honeypot (if applicable).
- *Log*: Whenever possible, the whole communication between the attacker and the sensor is logged, and can be presented to the user on demand.

ALERT DATA ANALYSIS

In this section, the results of the deployment of *TraCINg* for a five month observation period (March to July 2014) are presented. Initially, the details of the system's setup are given. Afterwards, the analysis of the alert data is conducted in a twofold manner. First, basic statistics are examined and discussed (see Section 6.3.2). Second, a similarity-based alert correlation algorithm is applied to the data and the results are discussed (see Section 6.3.3).

The purpose of this section is not to formally evaluate *TraCINg*. Rather, the driving idea is to highlight the richness of data that can be gathered by such a system as well as the depth of knowledge that can be gained by analyzing them. Complementary to the aforementioned reasoning, this section provides the reader with a discussion of "lessons learned" from the deployment of a fully functional cyber incident monitor for a long period of time. In fact, the results (particularly the ones related to correlated attacks) further motivate the need for collaborative intrusion detection. Lastly, various attack trends can be identified by such an analysis (e.g., the rise of IoT-related attacks).

System Setup

During the analysis period, data was collected from five different honeypots that were continuously monitoring and sending alerts to *TraCINg*. The specifics of the deployed sensors are summarized in Table 7. In more details, two sensors were located in Malaysia within a /24 sub-network. In addition, two sensors were deployed as cloud instances in USA within different /8 networks. Finally, one sensor was deployed in Greece.

Sensor Name	Country	Common Subnet Prefix
MY-01	Malaysia	/24
MY-02	Malaysia	
USA-01	USA	/8
USA-02	USA	
GR	Greece	-

Table 7: Sensor description and geographical information in TraCINg.

For the analysis of the data, the evaluation is restricted to the stationary deployed dionaea honeypots only. Data gathered from the mobile honeypots was not taken into account for two reasons. First, the number of users making use of the mobile honeypot, at the time of the analysis, was low and second the frequency of utilization of the honeypot was not consistent[4]. Thus, this could introduce bias into the analysis. Nevertheless, the study shows that even with a relatively low number of sensors, a large amount of distributed and correlated attacks can be detected.

Data analysis

In total, *TraCINg* sensors recorded 898,570 alerts during the examined period, from 30,101 distinct IP addresses, having their origin in 146 different countries[5]. It is interesting to note that attacks from USA and China alone, represented about 80% of the total number of alerts recorded by our system. Nevertheless, the conducted analysis suggests that this observation is not influenced by the geographical location of the deployed sensors.

4 A preliminary analysis of more recent, yet scattered, data gathered from the utilization of *HosTaGe* mobile honeypots, between March and May of 2015, is as follows. Approximately 1000 alerts were sent from four different countries (i.e., Germany, United Arab Emirates, Italy, and Columbia). At a glance, 62% of the attacks were shellcode injection, 24% were targeting the TELNET protocol, 6.6% were targeting HTTP, and finally 4% were MySQL brute-force attacks.

5 It is, however, assumed that the adversaries do not make use of IP address spoofing techniques.

Attack Description	Total Number of Attacks
Portscan	507,571
MySQL Attacks	156,960
Transport Layer Attacks	116,414
VoIP Attacks	84,641
SMB Attacks	30,239
MS SQL Attacks	2,325
Shellcode Injection	420

Table 8: Most popular attack types and the corresponding number of occurrences in a 5 month period of *TraCINg* deployment.

Port	Protocol/Service	Number of Attacks
135	RPC	24,667
139	NetBIOS	20,249
23	Telnet	11,058
80	HTTP	10,735
445	SMB	9,294
443	HTTPS	3,400
25	SMTP	2,558
21	FTP	1,658
110	POP3	1,153
143	IMAP	597

Table 9: Top 10 attacked ports and protocols in a 5 month period of *TraCINg* deployment.

Table 8, presents a summary of the most popular attack types along with the number of attack occurrences. The attack description column of the table depicts the specifics of the attacks. In this case *MySQL*, *MS SQL*, *VOIP*, and *SMB* refer to malicious activity specific to these protocols, e.g., a brute-force attack. Moreover, the *Transport Layer* describes cases in which the adversary connected to a port, but no further activity was possible due to honeypot-related limitations, e.g., dionaea not being able to handle the connection. With respect to dionaea's *VOIP* emulation capabilities [183], the majority of the detections were brute-force attacks and scans conducted with tools such as *SipCli*[6].

In addition, Table 9 shows the most targeted ports and protocols. A number of findings come as a result of this analysis. First, around 56% of the total attacks were portscans. This can be considered as some-

6 SipCli VoIP audit tool: http://www.kaplansoft.com/SipCli

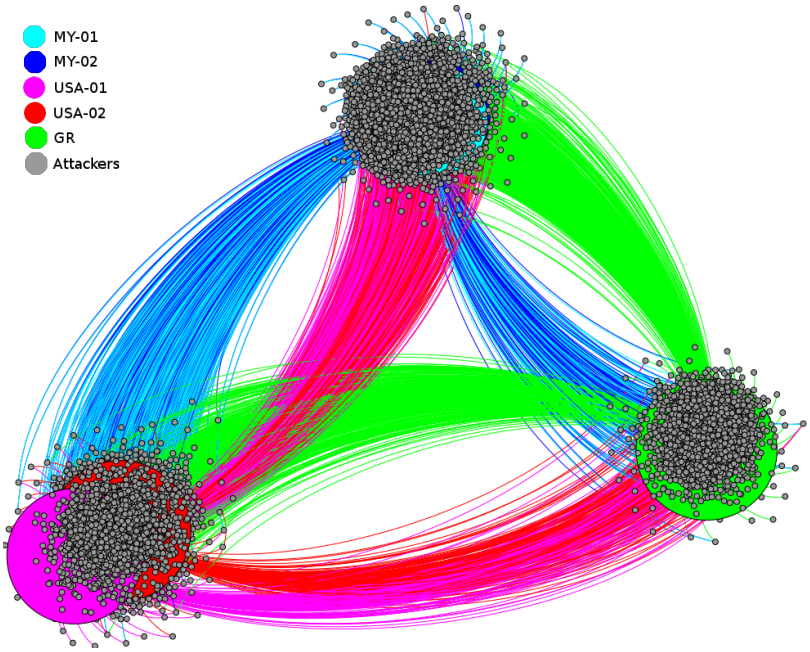


Figure 20: Graph representation of the alert data: attackers clustered close to their main targets and single-dimensional correlation (cf. Section 6.3.3.1) seen as edges connecting to neighbor clusters.

thing that is expected, especially with the rise of open source Internet-wide network scanners, e.g., ZMap [42]. In addition, it should be noted that bias can be introduced from such tools when utilized by researchers. Nevertheless, we consider this insignificant with regard to the overall analysis. Second, most of the detected MySQL attacks were brute-force attacks using default user-names and passwords. The most prominent one being a combination of *root* as a username along with a blank password. Moreover, Table 9 indicates that several attacks targeting Windows OS specific protocols and services, e.g., the *MS-RPC*, and *NetBIOS*, are still prominent. This also confirms that several old worm variants, e.g., Conficker [142], still hold an imposing position in the overall attack propagation scene. Lastly, a further analysis of the data suggests that a number of the attacks that were targeting the *Telnet* protocol were conducted by insecure/infected embedded devices, e.g., IP web-cams.

Correlation of attacks

Correlated attacks can be classified by differentiating between *single-dimensional* and a *two-dimensional* correlation of attacks, by following a *similarity*-based strategy [47]. *Single-dimensional* correlation groups attacks with the same origin, e.g., the same source IP address. Thus, it can be defined as the set of alerts originating from the same source IP address that target more than one sensor throughout the observed

period of five months. *Two-dimensional* correlation includes time as an additional parameter, i.e., it takes into account attacks that are observed within a specific time window and originate from the same adversary. The rationale behind the inclusion of time is twofold. First, a large portion of IP addresses is dynamic and hence adversaries may change IP addresses over time. Second, most of the attacks, nowadays, are conducted in a rather short-time frame.

Single-dimensional correlation

The analyzed dataset can be modeled as a directed graph $G = (V, E)$, where the set of nodes V represent all IP addresses involved in the detection process, i.e., both sensors and malicious users. The origin and the target nodes of an attack, are represented as a set of directed edges (u, v) with $u, v \in V$, that exist between the nodes.

Figure 20, is a representation of the alert dataset, that depicts two major findings: attack origins can be clustered, by vicinity, into three clusters and the single-dimensional correlation can be seen as the edges that connect to neighboring clusters. Specifically, the dataset was transformed to a directed graph with 30,106 nodes representing all distinct IP addresses (both sensors and malicious users), while edges correspond to the respective connections, i.e., adversaries connecting to the sensors. Figure 20 depicts, at a glance, an overview of the activities in our dataset.

The five different sensors (differentiated by distinct colors) converge into the three main clusters. The clustering is done based on the geo-location information of the sensors (cf. Section 6.3.1). As such, from the *five* deployed sensors, four of them were coupled into clusters of two. This is also explained in the system setup description in Section 6.3.1. The Malaysian sensors (within a /24 network) create tightly coupled clusters due to the high percentage of common attackers. In addition, the two sensors that are located in the USA, even though having distinct /8 networks, are also close to each other. Figure 20 also clusters the attackers based on the intensity of the alerts/attacks observed originating from them, i.e., nodes are placed closer to the sensors they attacked intensively. Moreover, *single-dimensional* correlation is observed when edges of a cluster (same color) are connecting to neighboring clustering groups, i.e., in the case when an adversary targets multiple sensors.

A similar analysis, from another perspective, on the ratio of *single-dimensional* correlation is shown in Figure 21. In more details, the figure depicts the relationship between the percentage of unique attackers and the targeted sensors. Almost 50% of the total number of attacks target at least two different sensors, during the *five* month period under investigation. However, a portion of this finding can be attributed to the 'closeness' of two sensors, i.e., sensors located within the same /24 sub-network. Nevertheless, as it is discussed in

the following section, even with the inclusion of time as a parameter, an almost continuous attacking behavior is observed on multiple sensors simultaneously.

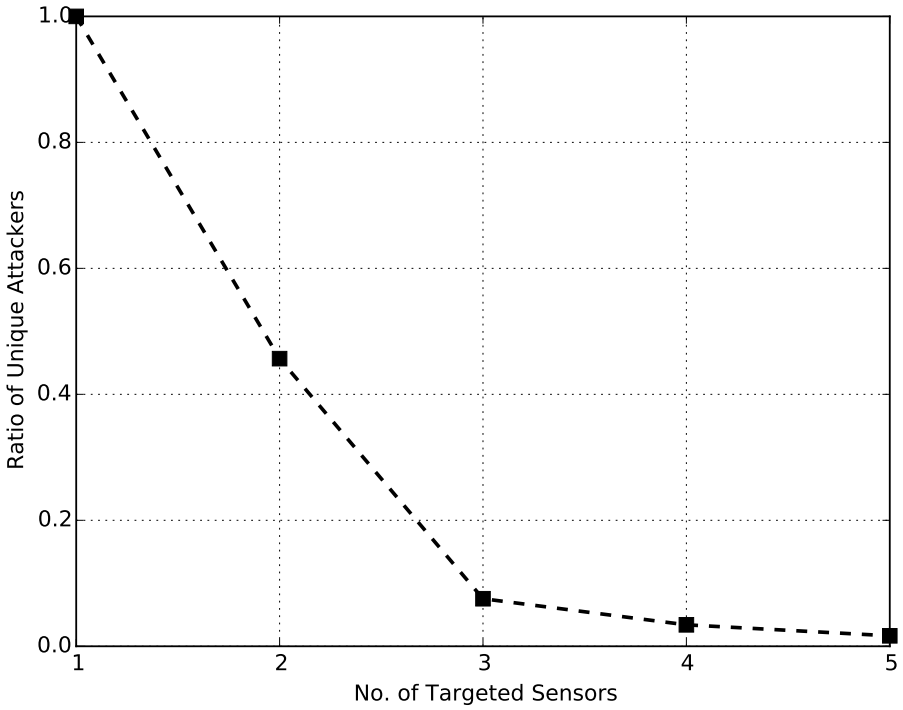


Figure 21: Ratio of unique attackers targeting multiple sensors in *TraCINg*.

Two-dimensional correlation

In *single-dimensional* correlation, the basis for the analysis is exclusively based on the source IP address of the adversaries. While this is reasonable, the time-frame in which attacks take place can also be taken into account. Such a *temporal-based similarity correlation* approach is based on the assumption that malicious behavior that is caused by the same adversary is likely to be observed in a short time-frame [138]. This approach comes with the advantage of being highly effective on detecting similar attacks, on connecting/aggregating attacks (and thus reducing the total number of alerts) and has been examined by several researchers (e.g., [2, 97]).

Figure 22, presents the number of unique attackers targeting multiple sensors within a short time frame. In more details, this two-dimensional correlation is measured with a sliding window of one hour (measurements taken every 30 minutes). The rationale behind the selection of these time intervals is that the system is utilizing honeypots, which assume any incoming connection as an attack. Hence, a single-dimensional correlation could detect attacks that are the result of regular probing (e.g., researchers that take Internet-wide measurements) and thus introduce false positives.

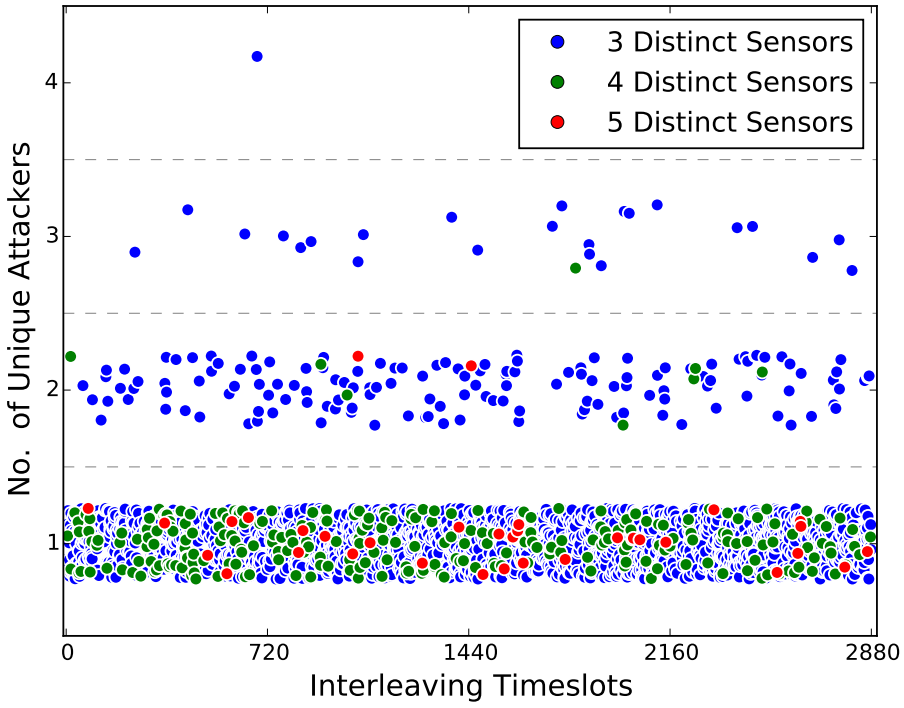


Figure 22: Unique attackers in *TraCINg* within a sliding window of one hour and with measurements taken every 30 minutes.

In fact, the 30 minutes time interval takes into account the state of the art in IPv4 network scanning. As of now, one of the fastest Internet-wide scanners, ZMap, is able to scan the IPv4 address space in approximately 45 minutes (via a random cyclic selection of IP addresses) [42]. Thus, with the chosen time interval the correlation algorithm attempts to reduce bias introduced by researchers' activities utilizing such mechanisms.

Note that for the sake of clarity, vertical jitter was applied to the points in the plot. Moreover, the figure was restricted to only two months (May to July 2014) of the overall period. Nevertheless, similar activity was observed for the removed time-frame. Throughout the observations, a pattern of attacking behavior can be identified in which, at least one unique attacker is targeting three or four different sensors (within a 30 minutes observation window).

Furthermore, there are also cases in which, up to four different adversaries targeted three sensors. Events like this, can indicate the discovery of a new vulnerability and thus the possibility of extensive scans over the Internet for exploitable insecure systems. The majority of two-dimensional correlation of alerts refers to portscans. In addition, for the cases of three and five overlapping sensors, also *MySQL* and *VOIP* brute-force attacks were detected.

SUMMARY

The previous chapters of this thesis already motivated the need for novel mechanisms to detect adversaries, and to generate and handle alert data. This chapter interconnected these two requirements of generating data and being able to handle them in a practical manner.

In more details, this chapter presented *TraCINg*, a cyber-incident monitor that is driven by honeypot sensors. The proposed system contributes to related work by offering an open source platform for studying cyber-attacks in a wide spectrum. *TraCINg* can assist researchers for examining alert data correlation algorithms and perform analysis of the alert data to identify novel attack trends and/or the propagation of malware. Moreover, the system provides a collaboration point for the *HosTaGe* mobile honeypot (cf. Chapter 5).

TraCINg also provides the ability to experiment in other areas of collaborative intrusion detection. In particular, as it will be shown later in this thesis, such a platform can be ideal for experimenting with probe-response attacks. Recently research has been conducted [143, 145, 27] that focuses on methods that can be utilized by attackers to successfully detect monitoring sensors that publish their findings publicly via the Internet. This thesis contributes in the area of probe-response attacks with the aid of *TraCINg*. The reader can refer to Chapter 10 for an in-depth discussion of probe-response attacks.

7

ID2T: AN INTRUSION DETECTION DATASET CREATION TOOLKIT

Chapters 5 and 6 dealt with the topic of alert data generation via the detection of genuine attacks. This chapter attempts to cope with the topic from the perspective of *synthetic* alert data generation.

In particular, this chapter discusses the topic of dynamic intrusion detection datasets generation by proposing ID2T, a concept and system for the creation of such realistic datasets. First, Section 7.2 proposes a number of requirements towards high quality datasets and dataset generation tools. Moreover, Section 7.3, provides an extensive discussion of the proposed approach by giving insights regarding the architecture, the attack generation mechanisms, and the GUI of the system. Section 7.4, presents the results of the evaluation of the ID2T prototype with a focus on the performance of the toolkit as well as the quality of the produced datasets. A further discussion of the outcome of the evaluation along with existing limitations of the system and further steps, are given in Section 7.5. Section 7.6 concludes this chapter. Finally, Figure 23 depicts the overview of the chapter with regard to the overall thesis structure.

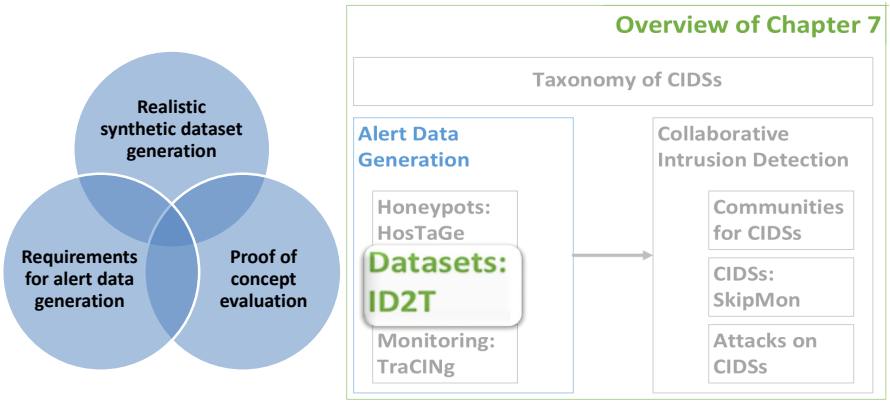


Figure 23: Overview of the Chapter and key contributions.

INTRODUCTION

In the previous chapters the thesis contributed in the area of alert data generation by introducing approaches that operate with a logic of detecting real attacks and afterwards gathering the respective alert data. However, such a logic is not always appropriate as it cannot provide a generic method for generating usable alert datasets. In this context, a common problem that researchers have to confront with, is identifying valid and commonly accepted datasets for evaluating their proposals [79].

In fact, this difficulty of discovering generally acceptable, comparable and publicly available alert datasets has not been tackled yet. That is, there is no commonly accepted dataset that one can use for the purposes of evaluating an intrusion detection algorithm or system. This generates several challenges when assessing novel work in the area of IDSs. First, in many cases researchers utilize significantly old datasets, e.g., the DARPA 1999 [92] dataset. Even though this dataset was considered a standard in the past, this is not the case any more as a lot of concrete work has criticized it. Moreover, in other cases many systems are evaluated with non-publicly available datasets. In such a case other problems arise, such as the reproducibility of the presented results.

This chapter proposes ID2T, a dataset generation toolkit that is intended for the evaluation of IDSs. As it is depicted in Figure 24, ID2T takes as input network traffic files (of an arbitrary size), injects network attacks and generates a labeled dataset. The toolkit is envisioned to be able to assist researchers to, out of the box, generate comparable datasets and hence publish acceptable and easily reproducible results. In addition, ID2T also aims on being practically used in real-world networks, e.g., corporate networks, for creating highly tailored datasets for the evaluation of the intended internal security mechanisms. In this context, the toolkit and the code for its prototype are being published publicly [118] so that the research community can be of benefit. The performance of the ID2T prototype is evaluated by first showing that the system can handle large amounts of network traffic as input. Furthermore, specific properties of generated datasets are examined to assess whether they contain artifacts.

REQUIREMENTS

This section proposes functional and non-functional requirements, for tools and their generated datasets, which aim at becoming useful for the evaluation of intrusion detection systems and algorithms. Lastly, the section also touches on the meaning and importance of *quality*, a non-functional requirement, in the context of creating datasets with automated tools.

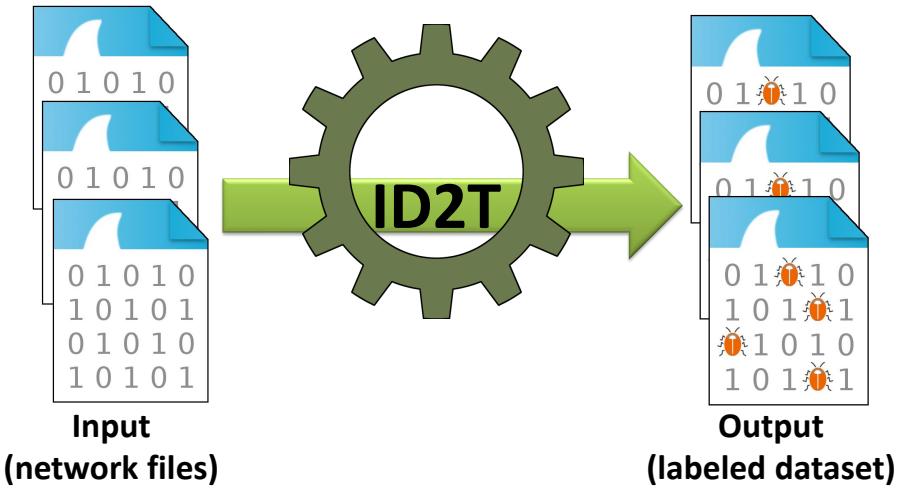


Figure 24: High level overview of the ID2T concept.

Requirements can be split into the functional and non-functional classes respectively. Both categories are discussed in the following. In addition, a special non-functional requirement, namely dataset *quality*, is further discussed in this section.

Functional Requirements

Functional requirements relate to certain practical properties of a dataset generation tool, its output dataset or both.

- Payload Availability: The payload of packets needs to be included in the generated dataset. This is important for its usability, as this information is often required for intrusion detection algorithms to work properly. Note that in many cases the payload is removed from datasets due to the need for anonymization.
- Attack Diversity: The dataset or tool must contain a broad range of attacks that span from traditional network attacks, e.g., portscans, to up to date and novel malicious activity, e.g., a sophisticated malware.
- Labeled Data: The generated dataset needs to contain clearly identified labels of the malicious traffic.
- Ground Truth: Besides labels the dataset should be able to guarantee the absence of attacks or anomalies in the labeled as non-malicious data. Usually this can only be achieved via a synthetic dataset generation.

Non-Functional Requirements

As the name implies, non-functional requirements refer to the properties of the generated dataset, the toolkit that influences the quality of the produced dataset, or both.

- Availability & Reproducibility: The generated datasets must be publicly available and hence allow the reproducibility of experiments.
- Scalability: The toolkit for generating datasets should be able to handle as input, and also produce as an output, network files of arbitrary size.
- Interoperability & Flexibility: The toolkit needs to provide the user with a method, e.g., templates or an API, for creating new attacks or modifying existing ones.
- Quality: In the case of synthetic dataset creation, the attack generation process is required to actively avoid introducing undesired patterns, or *artifacts*, outside the scope of an attack. This requirement is further discussed in the following.

Dataset Quality

Many intrusion detection datasets (see Chapter 2.3) have had problems due to the injection of inadvertent patterns which are easy to recognize by pattern recognition techniques. Several such artifacts have been identified which should be addressed whenever two unrelated network packet capture files are merged as one or when a packet capture file is altered. The following list of defects, or artifacts, is non-exhaustive and only reflects the main sources of problems that were identified during the conducted experiments or by examining the related work [106, 101, 102].

- TTL Value Distribution: Due to the different connectivity characteristics of individual networks, the distribution of TTL Values varies. The distribution of TTL values must be replicated to avoid creating easy avenues of detection for learning algorithms.
- Packet Capture Time Record: Depending on many factors, such as the bandwidth and the number connected hosts, packets captured in a network are recorded with different time characteristics. Bursty or constant packet rates should be imitated as well as the distribution of packet inter-arrival times.
- Packet Checksum: It is not sufficient to only modify desired values in network packet capture files. It is crucial to recompute checksums wherever appropriate.

- IP Address Distribution: Depending on the network, IP addresses are usually concentrated between specific address ranges. Injected or modified packets should use IP addresses from the same regions of concentration.
- Network Link Reliability: Due to the nature of networks, it is common to lose packets due to a saturated link or exhausted resources. It is important to identify these problems and replicate them when appropriate.

A further discussion of the aforementioned requirements is given in the evaluation section of the ID2T prototype (cf. Section 7.4) as well as in the discussion section (cf. Section 7.5). Furthermore, for a discussion of the related work in the area of intrusion detection datasets the reader may refer to Chapter 2.3

ID2T

This section first discusses the architecture of the proposed approach, and subsequently gives insights on the attack generation modules of ID2T. In addition, it provides with a brief description of the implementation of the ID2T prototype.

Architecture

Figure 25, depicts an overview of the architecture of the system. With respect to the core part of the toolkit, there are four internal modules that react to user input for creating labeled datasets: the *statistics*, the *packet splitter*, the *attack controller* and the *merger*.

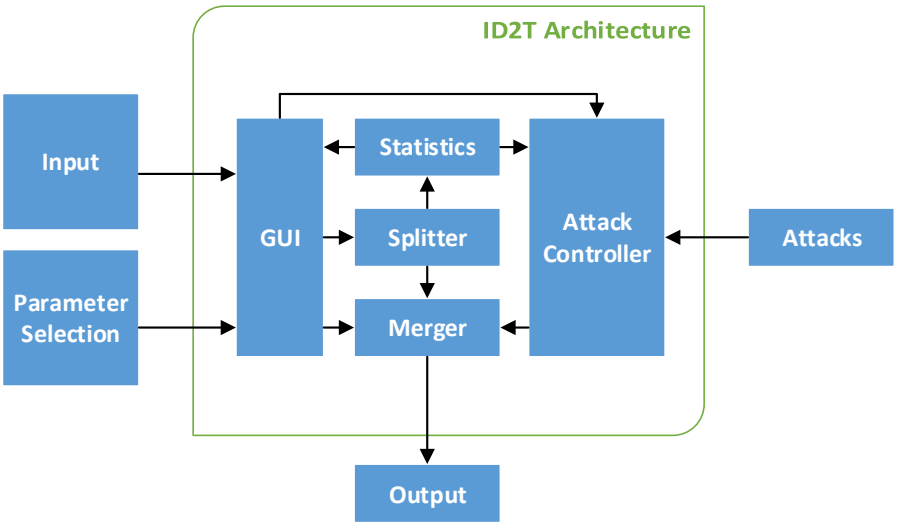


Figure 25: A high level view of the ID2T Architecture.

The *statistics* module is responsible for calculating statistics required by the attack controller module to replicate the quantitative and qualitative characteristics of the input. It is also connected with the GUI, in the sense of providing the user with an overview of the calculated information with regard to the input network file.

The *packet splitter* module is responsible for processing large network files by splitting them into smaller chunks, thus allowing an effective processing of network files. This splitting process can be achieved by utilizing two different parameters, seconds and packets. That is, the user can choose to split the input file with respect to a predefined parameter of seconds or similarly per number of packets. In addition, the module is envisioned to support various network file formats, e.g., pcap, pnpcap, and tcpdump.

Attacks are modeled utilizing a framework provided by the *attack controller* module. As this is one of the core parts of ID2T it is discussed in details in the following (cf. Section 7.3.2).

Lastly, the *packet merging* module receives the network packets created by the attacks and merges them with the previously split input files. The module also provides the user the flexibility to select between the various generated attacks. After the malicious files of interest are specified, the merger combines them with the input dataset. During the merging process all the packets are chronologically sorted and stored as a single file, thus, representing a realistic evaluation dataset; a labeled dataset, accompanied with useful visualizations for understanding the injections, is the final result of the process.

Attack Generation

ID2T is able to generate attacks by utilizing two different techniques, namely *script-based attack generation* and *pcap modification*. The first one aims on generating attacks, based on python scripts, in which all consecutive packets share similar parameters (e.g., a DoS attack). The pcap modification technique functions by modifying available or user-captured pcap files, e.g., traffic generated by a specific malware. In the following, more details are given for each technique.

Script-based Attack Generation

The main idea behind programmable generation is that many cyber-attacks can be essentially modeled as a large number of packets with similar parameters, that utilize the same protocol. For instance, in the case of a DDoS attack, such a large number of packets would exist that make use of a certain protocol (e.g., TCP), originating from a plethora of IP addresses, and targeting one IP address. Hence, many attacks can be modeled in such a way, and can be practically implemented in ID2T via the creation of the corresponding template (that includes the specification details of input parameters).

Moreover, implementing new attacks is a straightforward process; the only precondition is to follow a class-template style, that ID2T makes use of, and the toolkit will include the new attack automatically. However, attacks that include several protocols and a large amount of parameters and parameters cannot be effectively represented with such a technique. To cope with this, ID2T can utilize the pcap modification mechanism as described in the following.

PCAP Modification

In contrast to the script-based attack generation, the pcap modification offers a technique that is suitable for more complicated attacks. For example, this may refer to attacks that make use of multiple protocols, and/or include specific payloads. The first requirement for this injection technique is the existence of a pcap file that contains the traffic generated during such an attack. Such a case can be, for instance, a pcap file that includes the traffic generated by a specific malware. The user can either acquire such files from publicly available databases (e.g., from VirusTotal[1]) or create his own. When such a network file is available, the user can provide the system with specific parameters (e.g., the preferred malicious IP addresses) that will be used in the newly injected attack. Subsequently, ID2T makes use of certain functions, provided by the Scapy [14] network packet manipulation tool, to adjust parameters with respect to the user input. The outcome of the overall procedure is a new network file that includes the malicious traffic but with updated parameters. The merger module of the toolkit is able to take such a file as input and inject it to the original file given by the user.

ID2T Proof of concept

The prototype for ID2T has been implemented in Python. The GUI, as depicted in Figure 26, is responsible for visualizing all necessary elements of the ID2T.

In the current version of the prototype the procedure for generating a dataset follows three steps. First, the user specifies the input network file that will be used as the basis (ground truth) of the dataset generation. The statistics module will parse the data, will generate information with respect to the input and compute various parameters (e.g., the TTL distribution) that will be utilized in the next steps. Afterwards, the user can decide, via the attack generator, which attacks should be injected in the original network file. When this is decided and the respective parameters are given, the toolkit will generate an attack file. Note that the system is making use of information gathered from the first step to create realistic attacks. For instance, the

1 https://www.virustotal.com

TTL distribution of the network is taken into account so that the injected packets appear to be similar, while other information, such as the injection time, is properly randomized.

With respect to the labeling of attacks, the prototype is utilizing the MAC addresses as a marker. That is, the user can specify the MAC addresses of the adversaries to have the same unique value. Finally, ID2T will produce the final output, that is a labeled dataset in the form of a network pcap file. Afterwards one can easily identify the injected malicious behavior by checking the respective MAC address values.

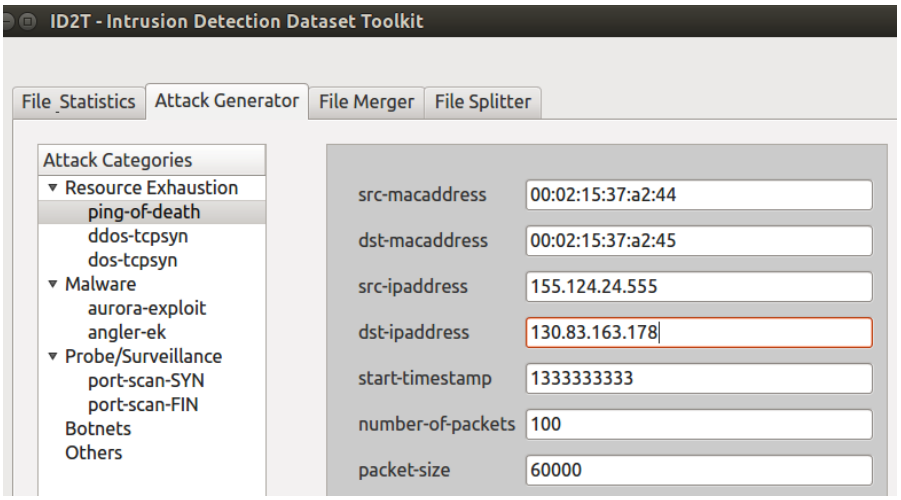


Figure 26: GUI view of the ID2T prototype.

With regard to the attack generation the prototype already implements several attacks. In more details, resource exhaustion attacks such as the *Ping of Death (PoD)*, a TCP-SYN DoS and a TCP-SYN DDoS have been implemented with the script-based attack generation technique. Similarly, a number of port-scan attacks are also offered. Furthermore, two malware attacks can also be utilized (representing the Angler malware and traffic generated by the Aurora exploit) by making use of the pcap modification mechanism.

EVALUATION

The ID2T prototype will be evaluated in a twofold manner. First, the performance of the toolkit is examined in terms of its ability to handle network files of arbitrary size. Afterwards, an attempt to evaluate the quality of the generated datasets is made, by examining common mistakes and criticisms of other synthetically created datasets as discovered in the related work.

Performance Evaluation

For an intrusion detection dataset toolkit to be practically *usable*, it is important that it can efficiently handle large network files as input. In the following, three different experiments are conducted that focus on the parameters and properties that can influence the performance of ID2T. In particular, the following measurements are emphasizing on the size of the network input file, the time required for the generation of packets (to be injected in the original file), and lastly the time required for the merging process (between the input file and the attack packets).

Figure 27, depicts the time required for the statistics module to perform its functions into large network files. For this, real-word traffic files were used, taken from the MAWI dataset [52]. The maximum utilized dataset size in the experiment, i.e., 13GB, corresponds to one day of traffic data from the MAWI dataset. Therefore, such a file size can be considered a reasonable upper bound for the usability of the toolkit. As one can notice, the toolkit is able to handle large datasets in a reasonable time window.

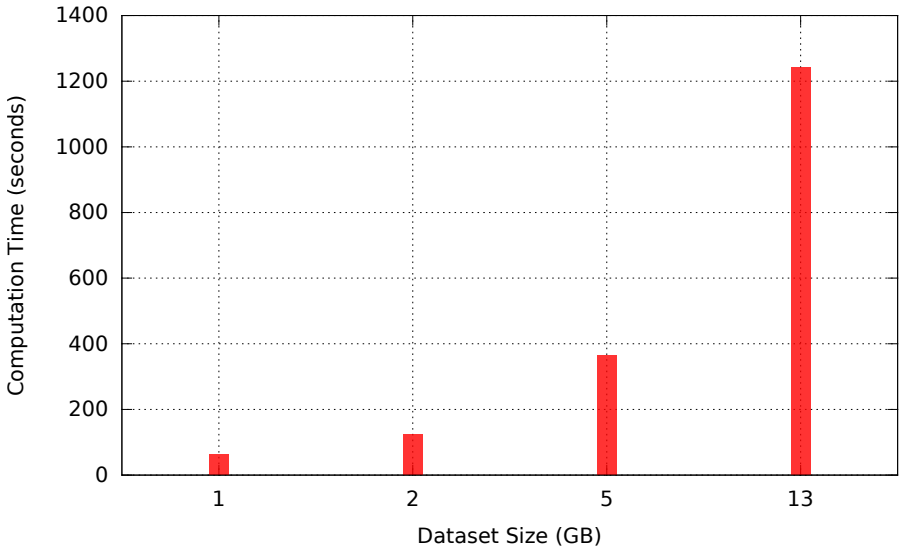


Figure 27: Performance of the statistics module for different dataset sizes.

Likewise, Figure 28 depicts the attack generation time required for the injection of attacks with respect to the number of generated and injected packets. In particular, the generation time, for the TCP-SYN DoS and DDoS as well as the PoD attack, is examined. The results suggest that the prototype is able to effectively perform the injection even when a very large number of packets is injected.

Furthermore, Table 10 shows four different experiments, each of them conducted with various input datasets (background data) and injected attacks. For the first three, a 2GB slice of the MAWI dataset [52] was used, while the last (fourth) assessment is based on 13.8GB

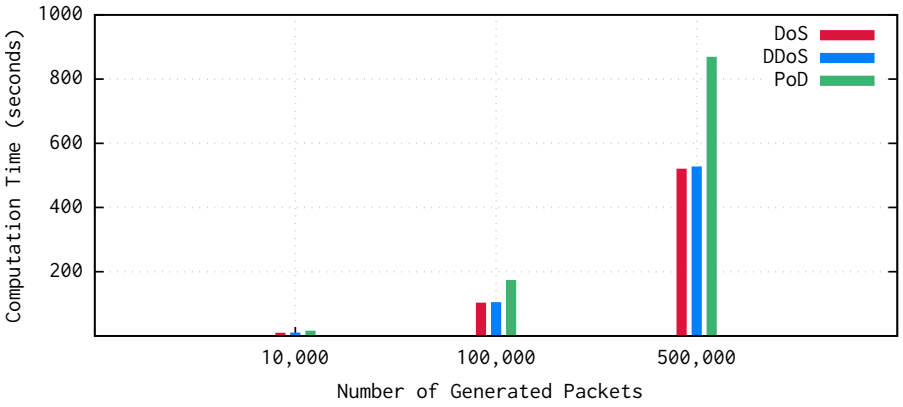


Figure 28: Attack generation time with respect to the number of generated packets.

of data from the same dataset. In the first experiment the 2GB background data is merged with 500,000 TCP SYN packets and 1,024 Port Scan packets. The second assessment is similar to the first one, but it adds additional 2,724 packets of the Angler malware. The last 2GB merging evaluation uses the four captures similar to the previous two scenarios, but also adds 6,704 packets of the Aurora exploit. Finally, for the 13.8GB input dataset all four previous attacks are merged and utilized. In all cases the timestamps of the attacks have been modified so that they fit the time of the input dataset. As a result of the merging process we observe a non-linear increase in the computation time, but still providing reasonable timing for such large network data files.

Input Dataset Size	TCP SYN (packets)	Port Scan (packets)	Exploit 1 (packets)	Exploit 2 (packets)	Total Time (seconds)
2GB	500,000	1,024	-	-	19.884
2GB	500,000	1,024	2,724	-	20.156
2GB	500,000	1,024	2,724	6,704	20.638
13.8GB	500,000	1,024	2,724	6,704	199.128

Table 10: Overall merging performance of ID2T for various datasets and attacks.

Artifacts Avoidance

It is of high importance that parameters that may introduce artifacts to the output dataset are properly generated. In the following, such parameters are examined along with a discussion on how ID2T attempts to avoid the generation of such artifacts.

TTL Distribution

It is important to take into consideration the TTL distribution of a network before injecting attacks into it. Hence, for examining the quality of the TTL distribution of ID2T and how realistic it is, the outcome of a dataset generated is examined when the input dataset is originating from the the MAWI dataset [52].

In more details, Figure 29 depicts a TTL distribution comparison of the MAWI dataset and the respective ID2T dataset when taking as input this dataset. As one can observe, the statistics module correctly derived the TTL distribution of the input file and hence the toolkit generated a dataset that closely follows such a distribution.

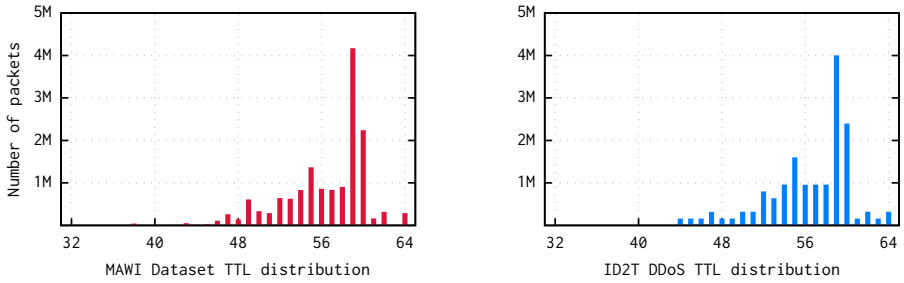


Figure 29: TTL distribution comparison of the MAWI dataset and ID2T.

Consecutive Packets Time

In a real world network, packets can take different paths which can be dictated by the routing information or the load of routing devices. Similarly, some packets might be dropped or lost and thus never reach their final destination.

Therefore, such network communication characteristics are examined, on the basis of ID2T, by modeling the time between two consecutive packets. Ideally, the time between consecutive packets (Δ time) should follow a burst behavior in which large number of packets are sent and respectively received in a short time range. Additionally, there should be distinct time periods with slight packet delays to mimic network congestions or other communication issues.

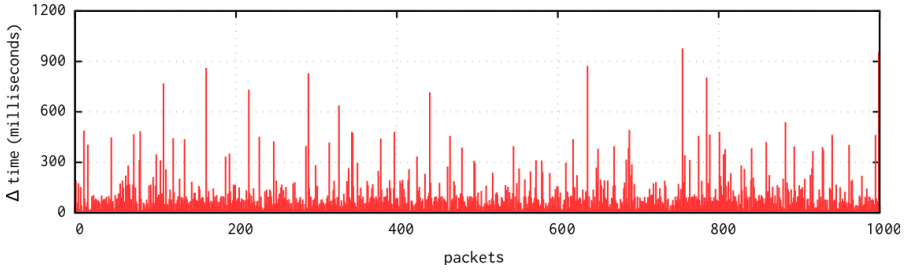


Figure 30: Modeling time between two consecutive packets with a 10 p/s rate.

Figure 30 depicts the case of the Δ time for 1000 consecutive packets of a TCP-SYN DoS attack generated by ID2T in which a packet rate of 10 packets per second is selected. This implies that the time distance between 10 packet timestamps should be approximately one second. To achieve this ID2T makes use of various randomization functions via its discrete probability distribution sub-module (inside the statistics module). Similarly, Figure 31 presents the same Δ time for a DoS attack but with an increase packet rate of 100 packets per second. Both experiments suggest that the consecutive packet time distribution has similar characteristics to the ones of a real network.

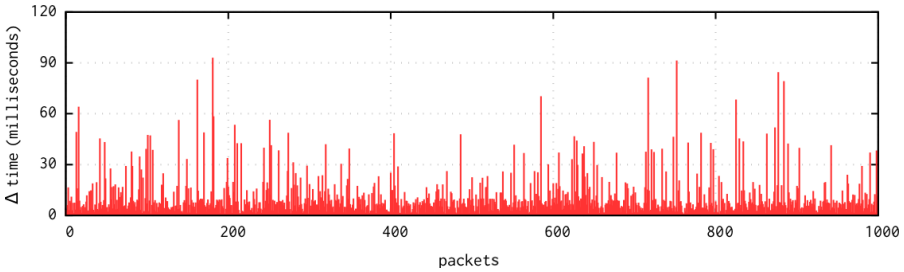


Figure 31: Modeling time between two consecutive packets with a 100 p/s rate.

Packet Head Checksum Calculation

The packet integrity of many protocols, e.g., IP and TCP, is protected by the usage of checksums. Such checksum calculations are important in the context of a synthetic dataset generator as inconsistencies can introduce significant artifacts. For instance, such a case was discovered by [102] with regard to the DARPA dataset and particularly with the tools utilized for generating attacks. As a result, in such a scenario anomaly detectors can differentiate between normal and malicious traffic by identifying such inconsistencies in the checksums.

To overcome such problems, checksum values are computed instantly when attacks are injected into a network file. In other situations such as pcap capture modification, where packet header information has to be forged according to user-defined parameters, another approach is taken. As the checksum is related to the packet's header data, any alternation in it will lead to changes in the respective checksum value. Therefore, ID2T recalculates the checksums of every packet that has been modified. In this context, packets containing incorrect IP and TCP checksums have been manually injected into pcap files. Afterwards, these files are given as an input to ID2T and attacks are injected. The output was thoughtfully checked with Wireshark [119] and did not contain any incorrect IP or TCP checksums.

IP selection and distribution

During the injection of attacks, the generation of source IP addresses, their distribution and randomization are of high importance to be able to create realistic datasets. ID2T handles this by first modeling the user input (i.e., the given pcap file), and suggesting proper values. In addition, the toolkit offers the ability to add different weights of occurrences for certain IP addresses. For instance, this might be utilized to illustrate more powerful attackers. Lastly, the toolkit excludes certain IP addresses from the selection process (e.g., an IP that starts with 192 must not be used as the source IP of a DDoS attack).

DISCUSSION

The performance and qualitative evaluation for the prototype of ID2T corroborated the hypothesis that ID2T is scalable and fulfills the requirements proposed in Section 7.2. The toolkit is able to handle large network files in a reasonable time. In addition, the examination of various parameters suggests that the generated datasets contain realistic properties. There are a few challenges that remain, however, in order to eliminate as many artifacts as possible from synthetically injected attacks.

First, there is a need for novel metrics to measure the quality of synthetically generated datasets. For instance, such a metric could be formally defined as a function that includes a weighted composition of all possible parameters that may introduce undesired artifacts. Second, a more in-depth investigation into the *quality* requirements, beyond the ones already identified in the related work, is needed in order to establish how the resulting datasets are affected.

Many functionalities of the ID2T prototype currently include steps that the user has to manually perform (for instance, using the results of the statistics analysis). A better interconnection and automation is planned for the process of selecting parameters from the statistics collected during the static analysis of the user packet capture files. Additionally, it is intended to further develop the attack controller module; focusing on producing a more flexible method of creating custom and novel attacks.

Finally, the merger module, along with the statistics and the attack controller, are envisioned to have an additional role. Cyber-attacks may introduce various changes on a network and currently this behavior is not fully reflected in ID2T. For instance, during a DDoS attack the number of dropped packets might increase significantly. There is a trade-off between introducing such changes in the network without inserting artifacts. This is considered as the next step for ID2T and will be examined in the future work.

SUMMARY

This chapter presented a novel approach towards the generation of synthetic, yet realistic, intrusion detection datasets. The chapter serves as a first step in the direction of modeling the prerequisites for the construction of high quality datasets. By studying the related work in the area, a comprehensive list of functional and non-functional requirements is proposed. In addition, specific properties that influence the quality of a dataset are discussed. Furthermore, the proposed system, namely ID2T, contributes in the area of intrusion detection evaluation. It offers a methodological approach for injecting network files with cyber-attacks to generate labeled datasets. Lastly, the developed prototype has been extensively evaluated with a focus on parameters and properties that might introduce artifacts.

The current chapter, along with Chapters 5 and 6, concludes the contributions of this dissertation in the area of alert data generation. The upcoming third part of the thesis will be presenting contributions in the core areas of collaborative intrusion detection.

Part III

COLLABORATIVE INTRUSION DETECTION SYSTEMS

The second part of the dissertation proposed contributions that aimed on the detection of attacks and the generation of alert data. The upcoming *third* part of this thesis presents contributions in the core area of CIDSs. First, Chapter 8 introduces the concept of *communities of sensors* accompanied by the utilization of ensemble learning techniques for CIDSs. On the basis of this, Chapter 9 proposes a novel *distributed CIDS* that exhibits a sophisticated correlation mechanism while also supporting the need for locality. Lastly, Chapter 10 contributes in the area of *attacks* in CIDSs by proposing improved techniques for detecting CIDS sensors as well as respective mitigation mechanisms.

8

COMMUNITY-BASED COLLABORATIVE INTRUSION DETECTION

The previous contributions of the thesis focused on generating alert data and on novel mechanisms for the detection of attacks. This chapter introduces and discusses the concept of communities in the context of collaborative intrusion detection. The main contribution, of the chapter at hand, is a CIDS concept that applies the idea of communities of sensors that collaborate by exchanging features of network traffic to create a holistic picture of the monitored network. The remainder of this chapter is organized as follows: Section 8.2 presents the proposed community-based CIDS concept; the problem is formalized, and the respective parameters and proposed algorithms are described in detail. Section 8.3 evaluates the community concept by applying it in an anomaly detection scenario. Section 8.4 concludes the chapter, and gives insights into further directions of the proposed approach. In addition, the latter part provides the reader with a logical interconnection towards Chapter 9. Finally, Figure 32 depicts the overview of the chapter with regard to the overall thesis structure.

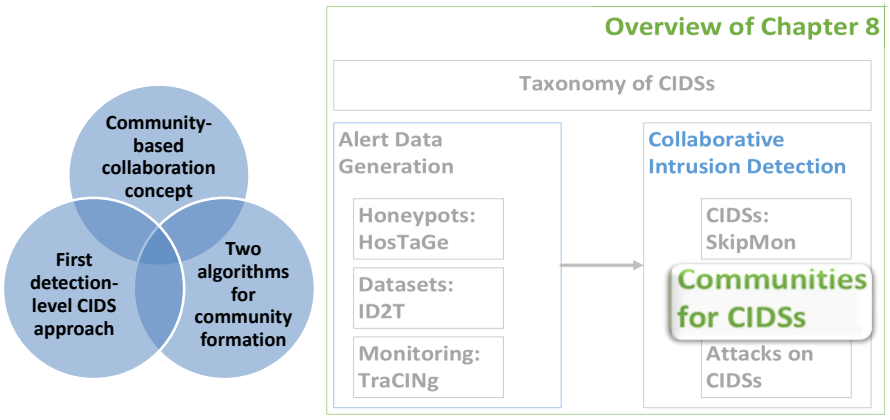


Figure 32: Overview of the Chapter and key contributions.

INTRODUCTION

The previous chapters provided contributions mainly in the areas of alert generation and attack detection. As discussed (see Chapter 2.1), the detection of intrusions with IDSs is typically performed through *misuse analysis* or *anomaly detection*. Misuse analysis assumes the availability of fingerprints of previously seen attacks, so that they can be detected upon their next occurrence. Anomaly detection establishes a model of normal system behavior. Each deviation from this model is an anomaly and thus a potential attack.

To create a holistic view of a monitored network, collaboration between IDSs is required, which has led to the development of CIDSs (see Chapter 3) that can be centralized or distributed, as discussed in Chapter 4. Distributed CIDSs provide better scalability than centralized CIDSs while reducing the communication overhead. However, compared to such systems, this usually comes at the cost of a decreased detection precision, i.e., the ratio between true alarms (or *true positives*) and the total number of alarms (*true positives* + *false positives*), as there is no component in the system with global information.

CIDSs exchange data either on the *alarm* or *detection* level. Information exchange on the alarm level, e.g., [22], encompasses the exchange of intrusion alarms for post processing. The main goal of this type of collaboration is to ease the manual task of analyzing all issued alarms by creating summaries and to discover related attacks. In contrast, collaboration on the detection level encompasses the exchange of monitored information (or data features) to collaboratively create or improve mathematical models. These mathematical models aim to improve the detection accuracy and, thus, lower the number of False Alarms (FAs). However, as discussed in Chapter 4, there is no CIDS that currently supports data exchange on the detection level [172]. This chapter takes into account the fact that, on the detection level, ensemble learning can be applied as a distributed machine learning method [122]. Ensemble learning has been demonstrated to be effective in the generic setting of improving anomaly detection [197].

This chapter advances the state of the art by proposing a CIDS concept for learning models of normality to detect network anomalies. In this context, this is the first CIDS proposal to support data exchange on the detection level. The focus is not to introduce a full-fledged CIDS, but rather to demonstrate the applicability of ensemble learning on intrusion detection in a distributed and collaborative setting. As such, the chapter proposes the establishment of communities of sensors that exchange data to build anomaly detection models and detect anomalies collaboratively.

In more details, a sensor is able to participate in multiple communities concurrently, which enables the applicability of ensemble learning techniques. Each sensor shares data with its communities,

so that subsets of the entire dataset are created. This allows each community to create an alternative hypothesis from each subset. Each hypothesis represents a particular interpretation of normal behavior and all hypotheses can be used together to determine whether arbitrary network traffic is normal or not. The proposed CIDS concept is evaluated with a modified version of the DARPA dataset [92] that reflects a distributed monitoring setting. The results indicate that a community-based CIDS approach performs better, in terms of detection accuracy and precision, than isolated IDSs in the task of learning models of normality.

COMMUNITY-BASED COLLABORATIVE INTRUSION DETECTION

This section provides insights into the proposed community-based CIDS. The novelty of the approach lies on the ability of the CIDS to exchange data on the detection level and on the insights of how such a community-based approach would function.

In the following, a description of the concept is given, accompanied by a formal model and a discussion on how the parameters of the formal model affect the properties of the CIDS. Subsequently, the community formation algorithms are described along with an examination of how the formed communities are used to perform intrusion detection.

Basic Concept

Sensors are grouped into communities to create samples of the network traffic all sensors are capable of observing. The samples are used to learn models of normality and perform anomaly detection. This idea is inspired by ensemble learning and guarantees the reduction of variance in the process of learning [99]. The overall outcome is an increased detection performance, in contrast to isolated sensors, and the reduction of communication overhead, in contrast to centralized systems.

In each community, one sensor becomes a *community head*. Community heads retrieve monitored data features from all other sensors in their community and perform intrusion detection. Upon detecting attacks, community heads forward alarms to a central administration interface where further correlation may take place. Selecting community heads can be done either stochastically or coupled to specific sensor properties such as their computational capabilities.

This chapter focuses on the detection accuracy and precision a distributed CIDS can achieve. The practical realization of such a distributed community formation is out of the scope of the current chapter. However, sensors could be grouped together into a P2P network using DHTs or P2P-based gossiping techniques [57]. Afterwards, tech-

niques like flooding can be applied on top of the overlay to establish communities in a distributed way. In this context, the reader can refer to Chapter 9 for such an example of a P2P-based CIDS that makes use of a generalized community-based approach.

Formal Model

The community-based CIDS overlay can be modeled as a graph $G = (V, E)$ where the nodes V represent computer systems capable of communicating between each other through an overlay communication links E that exist between them. Let $S \subset V$ be the set of intrusion detection sensors capable of collaborating among each other to detect attacks. Additionally, let $u \in V$ be a central administration interface responsible for collecting the alarms issued by all IDSs $s \in S$ and for generating intrusion reports. A community is a subset $\mathcal{C} \subseteq S$ of sensors, with $n_c = |\mathcal{C}|$ members. The set of all communities is $\mathbb{C}$, and the total number of communities is $n_t = |\mathbb{C}|$. Each community $\mathcal{C}$ has one sensor $s^\star_{\mathcal{C}} \in \mathcal{C}$ chosen as the community head; responsible for performing data analysis and intrusion detection. Every other member $s \in \mathcal{C}$ is connected by an edge $e = (s, s^\star_{\mathcal{C}}) \in E$ to $s^\star_{\mathcal{C}}$. Each sensor s is responsible for sending all features extracted from the data they collect to $\{s^\star_{\mathcal{C}} | \forall \mathcal{C} \in \mathbb{C} : s \in \mathcal{C}\}$, i.e., all other community heads they are connected to. The community heads of all communities are summarized in the set $\mathcal{S}^\star = \bigcup_{\mathcal{C} \in \mathbb{C}} s^\star_{\mathcal{C}}$. Each sensor $s \in \mathcal{S}$ may be repeated up to n_s times between different communities.

Fig. 33 shows three different examples of parametrization. The parameters specify how sensors s and community heads $s^\star_{\mathcal{C}}$ are grouped together. In Scenario 1, two communities are shown ($n_t = 2$). These communities have four sensors each ($n_c = 4$) and each sensor is allowed to be used only once ($n_s = 1$). Scenario 2 depicts three communities ($n_t = 3$), each having three members ($n_c = 3$), where the sensors are allowed to be repeated at most twice ($n_s = 2$). Lastly, Scenario 3 shows four communities ($n_t = 4$) with two members each ($n_c = 2$) where sensors cannot be repeated more than once ($n_s = 1$).

Parameters for Building Communities

When doing collaborative intrusion detection with communities, three dimensions can be recognized that influence accuracy, scalability and communication overhead. First, this section discusses the influence of the size of communities n_c and second, the number of communities n_t. These two parameters allow to model a centralized CIDS, a fully distributed CIDS, or communities. Third, the section examines the impact of the number of times n_s a single sensor can be part of different communities.

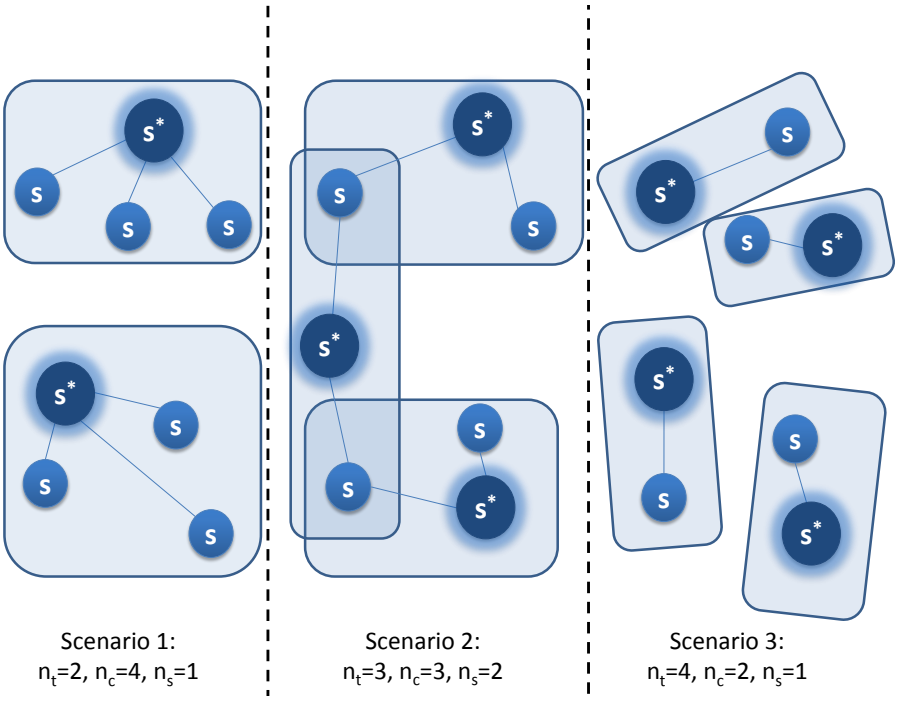


Figure 33: Example cases of two communities (left), three communities (center), and four communities (right), with sensors s and community heads $s^\star$.

Number of Sensors per Community (n_c)

The community size n_c significantly influences the detection accuracy. When $n_c = |\mathcal{S}|$, there is one community with all sensors. The sensor head $s^*_{\mathcal{C}}$ of this single community observes all data in the network and, thus, has full knowledge. This is equivalent to a centralized system that can access all data from one single location. In contrast, when $n_c = 1$, the scenario reflects $|\mathcal{S}|$ isolated sensors learning without any data being shared and no collaboration involved. In this scenario, each community has one sensor that must also be the community head. The size of n_c is bounded by $1 \leqslant n_c \leqslant |S|$.

The communication overhead affected by n_c can be expressed as the edges connecting the sensors $s \in \mathcal{S}$ to the community heads $s^\star \in \mathcal{S}^\star$; being inversely proportional to n_c. This overhead is calculated as $|S| - \frac{|S|}{n_c}$ and represents the number of edges required to interconnect all sensors to their respective community heads. Furthermore, with a small n_c, the system as a whole becomes more scalable as communities become responsible for analyzing less data. By increasing n_c, more information becomes available to each community head and a more accurate model can be derived; however, the communities become less scalable as more computational power and memory is required from every community head.

Number of Communities (n_t)

The second parameter that influences the detection accuracy and the precision is the total number of communities n_t. On the one hand, when $n_t = 1$, only one community is established. This is equivalent to $n_c = |\mathcal{S}|$. On the other hand, when $n_t = |\mathcal{S}|$ and $n_s = 1$, all sensors are their own community and no collaboration is involved. This is analogous to the scenario where $n_c = 1$. This shows that both n_t and n_c are inversely related to each other. The number of communities n_t is bounded according to $1 \leqslant n_t \leqslant |\mathcal{S}|$.

The parameter n_t affects scalability only in combination with n_c. Having a high number of communities does not imply anything unless n_c is taken into account. The main scalability issue in any distributed environment is the amount of data that needs to be collected and processed. For instance, a large n_t and low n_c implies that there are many communities processing small amounts of data.

Sensor Repetitions in Multiple Communities (n_s)

The n_s parameter is defined as the upper bound of the total number of times a sensor can be repeated in different communities. This parameter leverages the impact one specific sensor can have when communities are established stochastically. It is bounded according to $1 \leqslant n_s \leqslant n_t$. A sensor cannot be repeated within a community; otherwise, it would introduce bias because of the redundant data being shared.

As this parameter increases, more data is allowed to be repeated among many communities. The availability of all data can be augmented by increasing n_s. However, as this parameter increases, the communication overhead increases as well because sensors must transmit the same information to multiple community heads. The parameter n_s also directly affects the size of each community. As n_s increases, the number of sensors $|\mathcal{C}|$ of each community is increased on average. More members equates to more communication overhead.

Community Formation

The construction of communities demands criteria for coupling together the set of sensors S into communities $\mathcal{C} \in \mathbb{C}$. The coupling depends on parameters that affect how these are formed, i.e., the community size n_c, the total number of communities n_t, and the maximum sensor repetitions within different communities n_s. The remainder of this section contains a detailed discussion of coupling criteria and the algorithms that implement these criteria.

Coupling Criteria

One important design question of the proposed CIDS concept is how to assign sensors to communities, or, more precisely, how the data of all sensors is distributed for analysis. The proposed approach is inspired from the bagging ensemble technique. The bagging technique trains a classifier multiple times using different subsets of a dataset. Bagging reduces the variance of the detection accuracy [99]: it reduces the disagreement that might exist when communities are trained on different subsets of a dataset. To create different subsets of the data, data records are sampled with replacement from the entire dataset. To make a decision, every learner classifies the training dataset independently and a combination of all decisions is used to classify each individual training data.

The proposed community-based CIDS behaves like an ensemble of learners. Each community $\mathcal{C} \in \mathbb{C}$ is a classifier that learns with the data supplied by its members $s \in \mathcal{C}$. Sensors can appear in different communities, which is analogous to sampling batches of data observed by different sensors with replacement. The community size n_c specifies how much will be sampled. The number of communities n_t specifies how many classifiers will be built. Bagging does not usually limit the sampling in any way. Still, the n_s parameter is introduced, to limit the bias one single community may have in the whole system.

Ensemble methods traditionally split samples of the data randomly (with replacement) among the set of available learners. This is the motivation behind the stochastic creation of communities. Nevertheless, in the context of network data more intelligent decisions can also be used to split the data. For instance, network traffic can be split according to common network services, IP addresses or other network-related criteria. While this chapter focuses on stochastic community creation, the reader may refer to Chapter 9 for an alternative and more sophisticated (in terms of selection criteria) approach. At a glance, this chapter is trying to demonstrate how ensemble methods are able to perform well in the task of anomaly detection when coupling criteria are as general as possible.

Community Construction Algorithms

Multiple strategies can be used to form communities by varying the parameters n_t, n_c and n_s. Each parameter can be fixed to a specific value for all communities to share or vary for each individual community. Because of this, two different algorithms are proposed for constructing communities. Algorithm 1 fixes n_c to a particular value such that all communities exhibit the same size. The other two parameters, n_t and n_s, are left to vary for each community. In contrast, Algorithm 2 fixes the parameters n_s and tries to fix n_t whenever it is possible, while leaving n_c to vary for each community.

Algorithmus 1 : $\mathbf{comm_1}(\mathcal{S}, n_c)$

```
1  ℂ ← {∅}, T ← {∅}
2  for s ∈ S do
3      if s ∉ T then
4          C ← {s}
5          T ← T ∪ {s}
6          for |𝒞| ⩽ n_c do
7              s ← rand(S − 𝒞)
8              𝒞 ← 𝒞 ∪ {s}
9              T ← T ∪ {s}
10         ℂ ← ℂ ∪ {𝒞}
11     return ℂ
```

Given all sensors S and n_c as input, Algorithm 1 outputs a set of communities $\mathbb{C}$. This algorithm consists of two parts: In its first part (lines 2 - 5), the algorithm selects an initial sensor, not belonging to any other community, to start a new community. The list T is used to track sensors that already belong to a community. This restriction ensures that all sensors appear at least once among all communities while forming as few communities as possible. The second part of the algorithm (lines 6 - 9) adds random sensors to C from the set $\mathcal{S} - C$ until $|C| = n_c$.

Given all sensors S, n_t and n_s as inputs, Algorithm 2 outputs a set of communities $\mathbb{C}$ where $|\mathbb{C}| = n_t$ and no sensor is repeated more than n_s times among all communities. In contrast to Algorithm 1, this algorithm creates communities of different sizes. Equally to the n_c parameter of Algorithm 1, n_t has the property of generalizing how the community members collaborate as described in Section 8.2.3.2.

Algorithmus 2 : $\mathbf{comm_2}(\mathcal{S}, n_t, n_s)$

```
1  if n_s > n_t then
2      n_s = n_t
3  C_1, C_2, ..., C_{n_t} ← {∅}, {∅}, ..., {∅}
4  ℂ ← {C_1, C_2, ..., C_{n_t}}
5  for s ∈ 𝒮 do
6      x ← Uniform(1, n_s)
7      T ← {∅}
8      for 1 to x do
9          𝒞 ← rand(ℂ − T)
10         𝒞 ← 𝒞 ∪ {s}
11         T ← T ∪ {𝒞}
12     return ℂ
```

Algorithm 2 follows the following strategy. Lines 3 and 4 initialize the set $\mathbb{C}$ with n_t empty communities. The first loop of the algorithm (line 5) iterates over each available sensor $s \in \mathcal{S}$ to distribute it in the second loop (line 8). Each sensor s is placed, according to a uniform distribution in $[1, n_s]$, in multiple communities. It is possible that some communities are never chosen in line 9 and communities from the initial set $\mathbb{C}$ remain empty. Such empty communities are discarded.

Community-based Intrusion Detection

Each community $C \in \mathbb{C}$ represents an overlay where all sensors $s \in C$ are able to freely communicate with the community head, $s^*_{\mathcal{C}}$. All sensors $s \in \mathcal{S}$ extract features from the network they monitor and forward them to their respective community head where all these are bundled into one *aggregated training dataset*. Each $s^*_{\mathcal{C}}$ $\forall \mathcal{C} \in \mathbb{C}$ learns a model of normality using its aggregated training dataset, performs anomaly detection, and sends all resulting alarms to the central administration interface u. The unit u receives the alarms of all $|S^*|$ community heads, sorts the alarms by anomaly score, and reports the top-most anomalous alarms according to a predefined threshold.

After establishing a model of normality with the aggregated training dataset, the community heads perform anomaly detection using an *aggregated testing dataset* also gathered within the community. Sensors keep sending the same extracted data features used for creating the aggregated training dataset to the community head. However, the data features are now bundled into an aggregated testing dataset. The outcome of performing anomaly detection is the raising of alarms. Every community head sends these alarms to a central unit where alarm correlation and further analysis takes place.

EVALUATION

This section presents the results of detecting attacks in a modified version of the DARPA dataset using the novel idea of communities (cf. Section 8.2) coupled with the anomaly detection algorithm LERAD [103]. This evaluation demonstrates how communities outperform isolated sensors in the task of detecting intrusions using anomaly detection.

In the performed tests, the intrusion detection capabilities of centralized, isolated, and community-based CIDSs are compared. A community-based CIDS is a variant of centralized and isolated ones that represents a trade-off between scalability and accuracy. Each community analyzes the network traffic of multiple sensors and provides better scalability than centralized systems and better accuracy than isolated systems.

The DARPA Dataset

The dataset used for evaluation purposes is the DARPA dataset [92]. Regardless of this dataset having certain drawbacks (cf. Chapter 2.3), the dataset is used to compare the performance of three different systems under the same conditions; all of them utilizing the same labeled data. Moreover, the general availability/acceptability of this dataset and the precisely labeled traffic, without incorrect labels, makes this dataset more useful, in this particular context, than other alternatives such as the MAWILab [52], the CDX [139], or an ID2T-generated (see Chapter 7) dataset.

For the evaluation of the aforementioned approach, the DARPA dataset was modified to reflect the placement of multiple sensors at different points in the network rather than only at one. The description of how this is achieved is described in the following.

Modifications to the DARPA Dataset

The DARPA intrusion detection dataset [92] is a collection of network traffic obtained from a simulated military computer network with labeled attacks. In this evaluation, only the data records of incoming traffic are taken into account. There are a total of three weeks of training data and two weeks of testing data in the form of packet captures (pcap files). Only the third week of training data and both weeks of testing data are used. The training data does not contain attacks and is used to create models of normality. The testing data contains normal network traffic and 201 attacks ranging from denial of service to exploitation attempts. Due to the modifications described in the following paragraphs, 19 attacks are removed, i.e., traces of these attacks have been dropped as if no sensor was able to pick these up.

In the original dataset all network packets are captured by a single sensor at the ingress point of external traffic. For the purpose of testing the performance of multiple sensors analyzing the data independently of each other and within communities, the DARPA dataset is split according to the visible end-hosts in the local network. The incoming external traffic is split as if only end-hosts captured the traffic. The modified DARPA dataset emulates multiple sensors, each monitoring a single computer system, gathering data independently of each other. As a consequence, the original testing and training network traffic is split according to the local IPs found in the training set as if captured by multiple sensors instead of only one.

The DARPA modifications are illustrated in Fig. 34. The red sensor icons indicate the locations where network data is gathered. In the original DARPA dataset, one sensor, at the ingress point, collected all network traffic. The modified DARPA dataset emulates multiple sensors, each monitoring a single computer system, gathering data independently of each other.

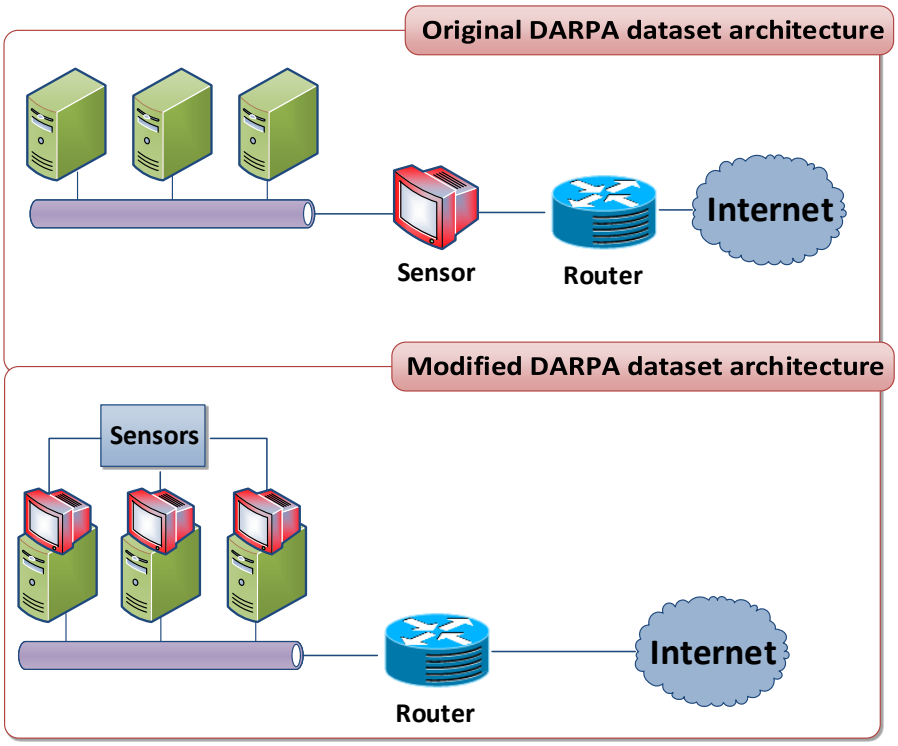


Figure 34: Modifications made to the 1999 DARPA Dataset.

Splitting the original dataset caused two important changes in the resulting dataset. First, all packets targeting an IP address of a non-existent endpoint in the local network are discarded as if no sensor would have seen these. The discarded packets were mostly generated by services that probed a large range of arbitrary IP addresses. Second, all packets targeting a local IP address in the testing dataset, targeted by incoming traffic that is not present in the training dataset, were also discarded. Many packets in the original testing dataset targeted local IP addresses not associated with normal traffic. Hence, for such traffic a model of normality cannot be derived. The end result was a training dataset containing 15 sensors (i.e., 15 different IP addresses).

The LERAD Integration

Rule learning algorithms, such as LERAD (see Chapter 2.1), are prime candidates for building ensembles of learners. An ensemble is a collection of classifiers that come together to classify novel instances as a group. Ensemble learning is comprised of a set of techniques to join the decisions made by different classifiers. The two most common techniques are called Bagging and Boosting [99]. Bagging is the process of sampling, with replacement, instances from a large dataset to create subsets. These subsets are used by many classifiers to learn

different models of normality (for anomaly detection). To classify a novel instance, each classifier makes a decision.

Multiple techniques can be used to mix all the classification decisions into one final decision. A popular technique is to consider each classifier output as a vote and use the class with the most votes. In this chapter, a technique is utilized in which the decision of the classifier with the most confidence in classification is used. The algorithm (see below) should be able to output not only the class, but also the confidence of detection as an *anomaly score*. Therefore, for one particular novel instance, the classifier with the highest anomaly score is taken as the classification decision.

To serve the aforementioned properties the LERAD [103] algorithm is chosen. LERAD is essentially utilized as the detection mechanism of all community heads $s^* \in S^*$. Each community head runs LERAD on its *aggregated training data* to learn rules that describe the network traffic of its community. These rules are the model of normality used for finding anomalies in the *aggregated testing dataset*. Records in this dataset are compared with the learned rules and the ones violating these are assigned an anomaly score. The rule violations, or alarms, are sent to the central administration interface u. The role of u is to collect and sort all alarms by anomaly score.

In the process of building the aggregated testing and training datasets, network traffic goes through pre-processing to extract 23 features which are effective for LERAD [103]. For each observed TCP stream the following features are extracted: the date and time; the destination and source address; the destination and source port; the duration of the TCP stream; the TCP flags of the first, second to last and last packets of the TCP stream; the byte length of the stream; and the first 8 words of the stream.

Experimental Setup

The main purpose of the evaluation is to measure the *accuracy* and the *precision* of detection. Accuracy is defined as the total number of attacks detected over the total number of attacks. The precision equates to the true alarms (or true positives) over the total number of alarms (true alarms + false alarms). Due to the stochastic nature of LERAD, each experiment is repeated 500[1] times and the presented outcome is the average of the accuracy and precision for all runs. The confidence intervals of these measurements are omitted in the figures, except for Figure 8.35(a), as they are insignificant.

1 The stochastic nature of the algorithm requires an adequate number of repetitions. The utilized number was chosen as an upper bound as no difference was observed to the results beyond this bound, i.e., the values already appear to converge and the confidence intervals are insignificant.

The detection accuracy and precision are measured using the alarms the central administrator interface u receives from all community heads. In a pre-processing stage, duplicated alarms within a time-frame of 60 seconds are removed as, according to the original DARPA competition, alarms are deemed true if they detect an attack within 60 seconds of its occurrence [92]. Each alarm is being analyzed, from highest to lowest anomaly score, assessing if the alarm is a true or false positive. This process continues until a predefined number of FAs is reached and all remaining alarms are discarded. Note that the procedure described above closely follows the method for evaluating LERAD as described in [103].

In every experiment, the accuracy and precision is examined with different numbers of random communities. Precision is defined as the ratio between true alarms (or *true positives*) and the total number of alarms (*true positives* + *false positives*).

Three cases can be distinguished given the size of the community:

- **Centralized System ($n_c = 15$):** All sensors send the extracted features to a single community head.
- **Isolated System ($n_c = 1$):** A community for each sensor ($|\mathbb{C}| = 15$) on its own without any cooperation.
- **Communities ($n_c = x \mid 1 < x < 15$):** Variable number of communities.

On the one hand, the community of size 15 is expected to outperform all others, in terms of detection accuracy and precision, given that all the features extracted are available in one single location for analysis. On the other hand, it is expected that 15 single independent communities will perform the worst overall as there is no collaboration involved. The following section shows that as communities include more sensors, the detection accuracy and precision is improved while at the same time leveraging the communication overhead. In addition, it is demonstrated that under certain conditions the communities achieve a detection precision similar to the centralized system with a better detection accuracy.

Results

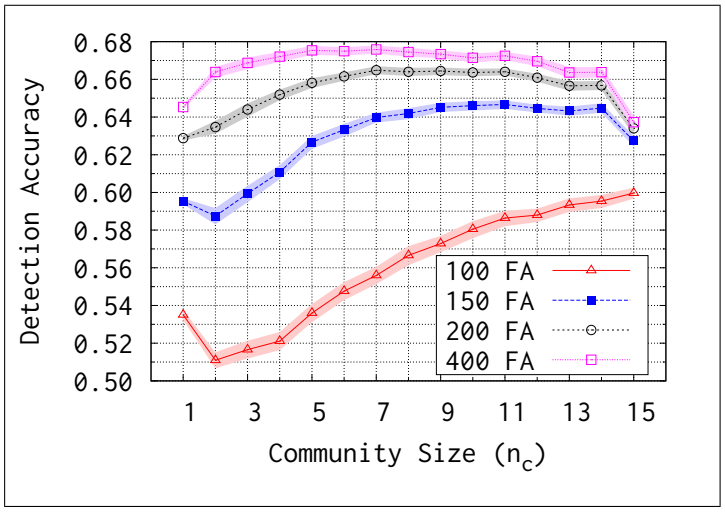
The analysis baseline is shown in Figure 35, where the detection accuracy and precision is compared for every possible community size n_c, as built by Algorithm 1. Figures 35 (a),(b) show the outcomes of the experiments by varying the FA limit, i.e., changing the threshold for raising alarms. Each anomaly detection experiment is carried out until a predefined number of FAs are issued. At this point, the detection is stopped and the results are recorded. The detection capabilities using 100, 150, 200 and 400 FAs are being measured. The testing data

corresponds to two weeks (10 total days) of data; as such, 100 FAs equates to an average of 10 FAs per day, 150 to 15 FAs per day, and so on. The shaded area around the solid lines in Figure 35 (a) shows the confidence intervals of the measurements.

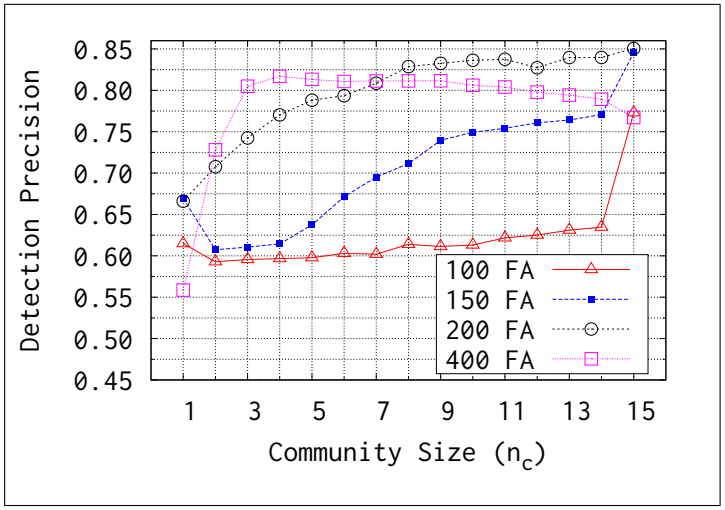
After 100 FAs are found in the sequential analysis of each alarm, from highest anomaly score to lowest, the accuracy and precision of the detections are reported. Figure 35 (b) shows that as communities grow in size, the precision is improved. This translates to the hypothesis that the centralized system would have the highest accuracy and precision rates. As seen in both Figures 35 (a),(b), if the 100 FA restriction is relaxed, some community sizes are able to improve the detection accuracy in contrast to the centralized system (when $n_c = 15$). At 200 FAs, most community sizes have better detection accuracy than the centralized system. In addition, relaxing the FA restriction allows the detection precision to converge to the one of the centralized system. Lastly, at 400 FAs, a point is reached where every community is able to outperform, in terms of accuracy, both the individual approaches as well as the centralized system. It should be noted that above the 400 FA limitation, no significant changes are observed. However, as seen in Figure 35 (b), the precision drops as the FAs are increased. With the 200 FAs limitation, communities with $n_c \in [9, 11]$, quickly approach the precision ratio of the centralized system.

The number of repeating sensors (n_s) has also some interesting properties that impact the detection accuracy of fixed community sizes. Figure 36 (a) shows the experiments of varying $n_s \in [1, 5]$ with Algorithm 2. The graphs being plotted show the impact n_s has on the detection accuracy with respect to the number of communities n_t. As more sensor repetitions are allowed, the overall accuracy is improved. Here it is also evident that the centralized system ($n_t = 1$) still outperforms all others. Furthermore, Figure 36 (b) strengthens the aforementioned statement that as n_t increases, the impact of n_s decreases.

To sum up, the experimental results indicate a number of interesting facts. First, as expected, a centralized architecture outperforms all others when the threshold for raising alarms is set high, i.e., when the number of FAs is constrained to low values. Nevertheless, communities provide fair detection and precision ratio and better communication overhead in comparison to a centralized system, while already outperforming individual IDSs at the lowest tolerated FA limit of 10 average alarms per day (100 FAs). Isolated sensors perform no collaboration and, in consequence, create less accurate models of normal traffic than the ones created by collaborating communities. Second, as the threshold for raising alarms is lowered (allowing 200 or more FAs), communities start to perform similarly to the centralized system; finally being able to outperform it (in terms of detection accuracy). This performance can be explained by the fact that, due to the



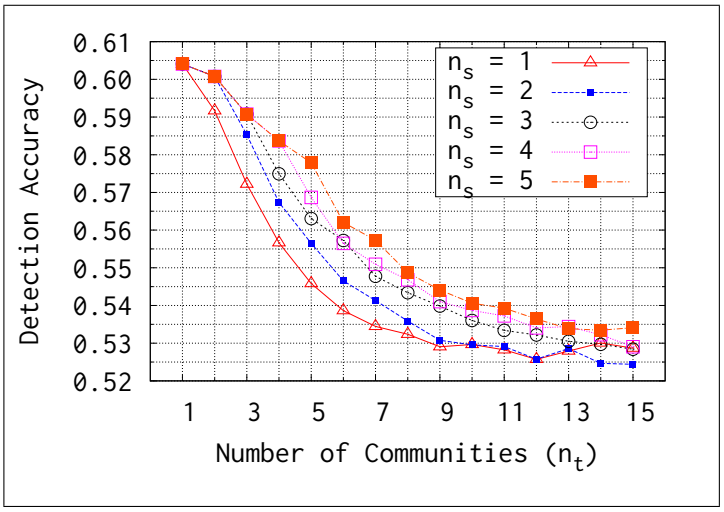
(a) Detection accuracy evaluated at different False Alarm (FA) rates.



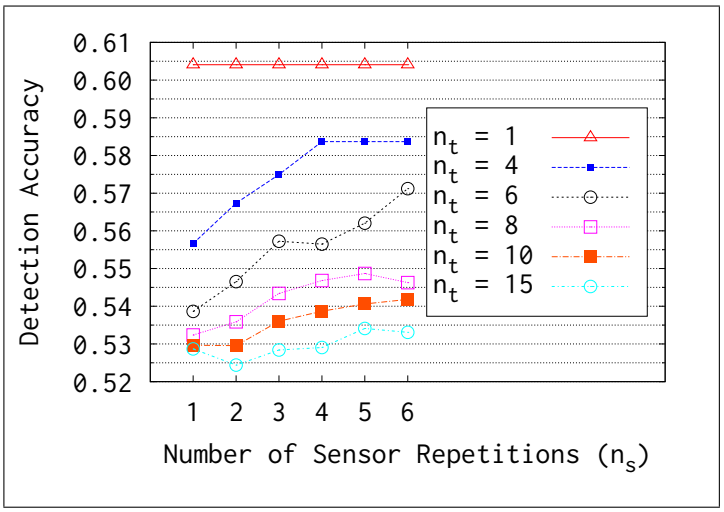
(b) Precision of detections at different False Alarms (FA).

Figure 35: Detection accuracy and precision at different FAs when communities are built using Algorithm 1.

stochastic nature of the proposed algorithm, there is a point where communities are able to gather together enough sensors to generate accurate enough models of normality that explain general network traffic patterns. In addition, these results also comply with the initial argumentation that the proposed community-based CIDS has properties similar to ensemble learning. The CIDS approach is able to improve performance by using different models of normality learned by different communities. Overall, the results in Figure 35 indicate that it is possible to find a combination of parameters n_t, n_c, n_s and a particular threshold for raising alarms that enables communities to perform close to a centralized system while reducing the communication overhead.



(a) Detection accuracy depending on the number of communities n_t evaluated using different repetitions n_s.



(b) Detection accuracy depending on the number of sensor repetitions n_s.

Figure 36: Accuracy when the communities are built using Algorithm 2.

For a production system, a practical scenario that can be envisioned for such a task can be the utilization of ID2T (see Chapter 7) with network traffic from a network, that is to be monitored, as input. In such a case a similar to the aforementioned evaluation can be performed to identify the optimal values for the community-related parameters.

For the particular instance of the modified DARPA dataset, the best results are given when the community size $n_c = 9$, the repetitions $n_s = 3$, the total communities $n_t = 4$ and the FA threshold is set to allow 200 FAs.

SUMMARY

The continuous sophistication of network attacks urges the development of novel IDSs and architectures. This chapter contributes in the area of collaborative intrusion detection by proposing a CIDS concept that applies the novel idea of communities of sensors that collaborate by exchanging features of network traffic to create sufficiently accurate normality models for performing intrusion detection. Moreover, a further contribution of the chapter at hand is the proposal of the first CIDS that supports alert data exchange on the detection rather than the alert level (cf. Chapter 4 and Section 8.1). In addition, two stochastic algorithms were developed that group sensors into communities and demonstrate how these communities are able to influence the detection capabilities and communication overhead of CIDSs. The experimental results indicate that the proposed community-based CIDS concept, performs better than isolated systems in terms of detection accuracy and precision. Furthermore, it has been demonstrated that communities can perform similarly to centralized systems even though less information is distributed to build normal models for anomaly detection and, as such, less communication overhead is involved.

Furthermore, the aforementioned contribution of the community-based CIDS approach is the basis on which chapter 9 is constructed. That is, the generalized idea of communities is utilized in a fully fledged distributed CIDS. In that sense the following chapter can be additionally seen as a practical realization (cf. section 8.2.1) of such a distributed community formation, comprising with additional sophisticated criteria for forming communities.

9

SKIPMON: A DOMAIN-AWARE COLLABORATIVE INTRUSION DETECTION SYSTEM

Chapter 8 introduced the concept of communities for collaborative intrusion detection. The chapter at hand adopts the generic idea of communities and builds a fully fledged system on top of it. In particular, this chapter proposes *SkipMon* a distributed CIDS and it is organized as follows. Section 9.2, provides an extensive description of the architecture of the system. Section 9.3 gives insights with regard to the implementation. Subsequently, Section 9.4 presents and discusses the results from the evaluation of *SkipMon*. Section 9.5 provides an overview of the section. Finally, Figure 37 depicts the overview of the chapter with regard to the overall thesis structure as well as the key contributions of the chapter..

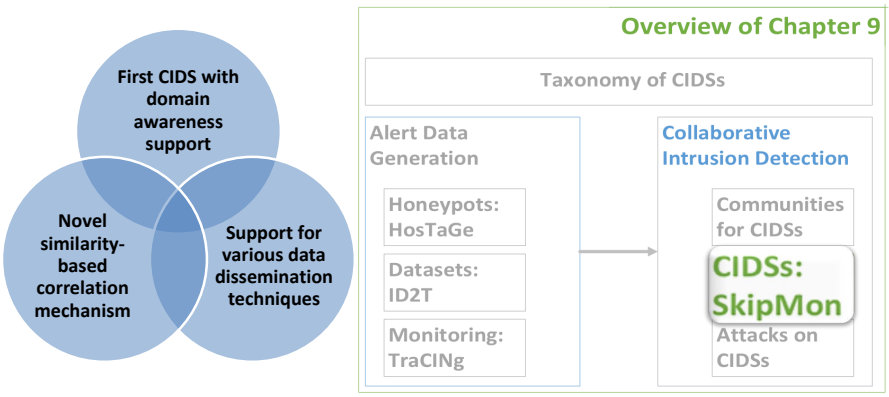


Figure 37: Overview of the Chapter and key contributions.

INTRODUCTION

This chapter adopts and advances the idea of communities of sensors that collaborate by exchanging alert data, which was presented in Chapter 8. In particular, the chapter adopts the generic communities concept but offers a more sophisticated method for correlating alerts and hence for constructing communities. Furthermore, the whole approach is examined as a fully functional CIDS.

For a CIDS to be efficient and usable in a practical manner, a number of requirements must be fulfilled. In the following, we recapitulate these requirements and emphasize on one of them, i.e., the domain awareness. The reader can refer to Chapter 3.2 for a detailed discussion of the requirements and to Chapter 4.2 for a qualitative comparison of the related work.

First, the CIDS has to provide *scalability*, i.e., support for the monitoring of arbitrary network sizes. This should also be accompanied by a minimal message *overhead* as well as by the *privacy* of the exchanged alert data. Furthermore, such a system needs to be able to control the flow of alert network traffic in such a way so that only sub-networks that are allowed to communicate, can exchange messages. This requirement, namely *domain awareness*, refers to the ability of the CIDS to constrain alert dissemination, to certain sub-domains of a network, with respect to the ongoing security policy of a corporation.

To better understand the necessity of such a requirement, the reader can consider the case, of a corporate network, in which different sub-networks are logically separated due to a strict security policy. For instance, the sub-network of the economics department may not be allowed to communicate with the development department, and so forth. This *domain awareness* property, to the best of the knowledge of the author (cf. Section 4.2.4) has not been addressed, so far, in the related work of CIDSs. However, this is significantly important for the practical realization of such systems.

This chapter presents *SkipMon*, a novel distributed CIDS approach that utilizes the SkipNet [67] P2P overlay for the basic communication of its monitoring sensors. *SkipMon* offers two major contributions in the area of CIDSs. First, via the utilization of SkipNet (cf. Appendix B), it supports domain awareness, i.e., the ability to, on-demand, constrain the dissemination of alerts to certain sub-domains of the monitored network. Furthermore, a novel mechanism is proposed for disseminating alert data and subsequently correlating the received information on the basis of bloom filters. In *SkipMon*, sensor nodes exchange (alert) network traffic to discover others that experience similar traffic patterns. For this, a compact (low-overhead), privacy-preserving data dissemination mechanism is proposed via the utilization of bloom filters. In particular, sensor nodes that experience similar traffic subsequently create communities of nodes for

exchanging more fine-grained alert data. In addition, *SkipMon* scales to large networks and is open-source, offering one of the first real-world implementations of a distributed CIDS [48].

The system is evaluated via the usage of real-world network attack traffic to determine the messaging overhead, the accuracy of the suggested communities, as well as the effectiveness of the respective domain awareness mechanisms. The conducted experiments indicate that *SkipMon* provides good accuracy rates into selecting the correct sensor nodes that experience similar alert traffic. To evaluate this we compare *SkipMon* to a centralized system with full knowledge of the alert data of all the participating sensors.

SKIPMON SYSTEM ARCHITECTURE

This section provides a detailed description of the *SkipMon* system by discussing its subcomponents. We utilize the architecture shown in Figure 38 to construct the five main building blocks that compose our system. The architectural design of *SkipMon* is inspired from the CIDS taxonomy as proposed in Chapter 4.1.

In more details, the *Local Monitoring* is responsible for the local detection as performed by the IDSs of each sensor. Sensors communicate by utilizing a P2P *membership management* protocol, i.e., the *SkipNet overlay*. Subsequently, sensors can exchange alert information by utilizing the *alert dissemination* mechanisms. In *SkipMon* a similarity-based *alert correlation* technique is utilized to identify sensors that experience similar traffic patterns. By making use of such a mechanism it is possible to detect distributed port scans as well as malware propagation. Each sensor learns the traffic patterns of others and it is able to utilize a *community formation* algorithm. Afterwards, sensors can exchange more fine-grained alert information only with their community members. This can be of benefit in terms of reducing the alert traffic of the CIDS. In the following subsections we detail each building block and how *SkipMon* fits in each block.

Local Monitoring

A CIDS utilizes several IDSs to monitor an entire network. *SkipMon* is envisioned to make use of standard IDSs, e.g., Snort [135] and Bro [120], as long as they support standardized alert formats, e.g., the IDMEF [39]. The current prototype of *SkipMon* (cf. Section 9.3) assumes the existence of such IDSs. In that sense, *SkipMon* is monitor agnostic; the only prerequisite is the input of alert data giving the system the ability of creating and disseminating local alert knowledge to other members of the CIDS.

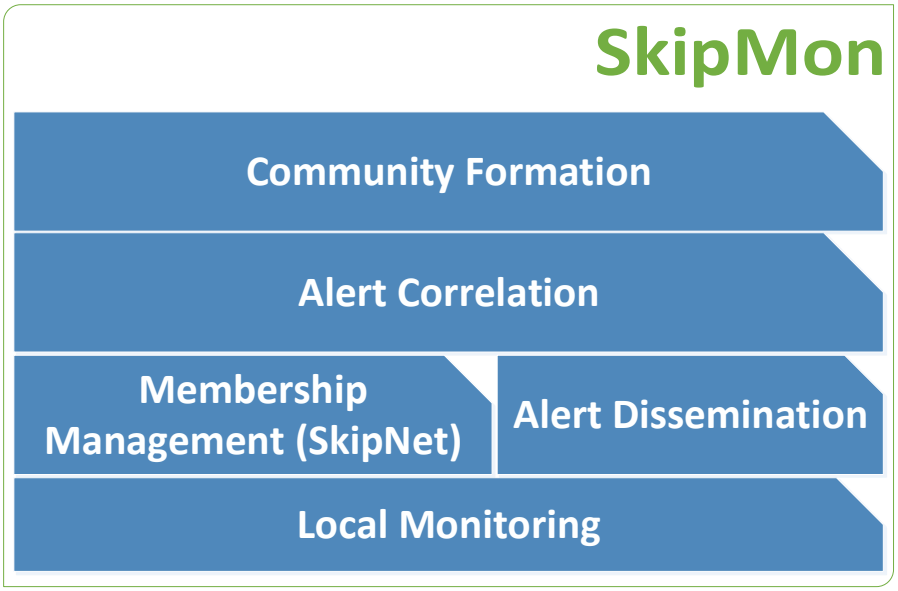


Figure 38: High level architecture of *SkipMon*.

SkipNet Overlay

For our system we make use of the SkipNet P2P overlay (cf. Appendix B). As discussed in the appendix, SkipNet can provide *data locality* and *domain awareness*. In this work, the focus lies on the domain awareness to share information only with authorized sensors of a monitored network. Therefore, for *SkipMon*, data will be forwarded between the nodes (monitoring sensors) of the system instead of storing it at a given node or set of nodes. Due to the routing algorithms and ordering of nodes in SkipNet, data that shall be exchanged in one domain can and will only be routed through nodes of this domain. This leads to implicit data locality, as data transferred between two nodes in the same domain will never leave the boundaries of the domain in transit.

Alert Dissemination

To keep the communication overhead in the system low, alert information needs to be disseminated efficiently. Therefore, we examine three alert dissemination techniques: *flooding*, *partial flooding*, and *gossiping*. Flooding is a common dissemination mechanism that has the advantage of a guaranteed reach of nodes (with a trade off of a high communication overhead). Partial flooding is inspired by flooding but takes into account the domain awareness requirement. Lastly, gossiping is a probabilistic approach for disseminating information into the CIDS.

Flooding

As the name implies, this alert dissemination mechanism operates with each node sending messages to all their neighbors in the SkipNet. Respectively, a node that receives a message will forward it to all of its neighbors. To minimize the overall communication overhead, redundant messages (i.e., messages that have been received from another node or path) are dropped. This is achieved by utilizing the message ID as we discuss in the next sub-section.

Finally, it should be noted that as flooding disseminates data to all nodes, regardless of their subnetwork, the domain awareness requirement cannot be fulfilled. Nevertheless, flooding can be a useful mechanism for disseminating important information inside the CIDS. For instance, such a mechanism can be utilized for informing all the monitoring sensors of the system for a detected ongoing attack.

Partial Flooding

In order to enable domain awareness, the system utilizes a partial flooding mechanism. Instead of exchanging messages with all possible neighbors, nodes selectively exchange messages only with neighbors of the same sub-domain. This is made possible due to the fact that each message contains a domain awareness value (cf. Section 9.2.4.1) which can be used to query for neighbors in the same domain level. Such a dissemination technique is particularly useful when security policies exist that prohibit the communication between different domains of a network.

Gossiping

Flooding creates significant network overhead that might exceed the available bandwidth and computational capabilities of CIDSs' sensors. In this context, the gossiping algorithm proposed by Kermarrec et al. [77, 76] is adapted and utilized in *SkipMon*.

The original algorithm uses a hierarchical communication approach where nodes are grouped into clusters. The communication links between nodes inside the same cluster are called *intracluster* links. The communication links that nodes within one cluster maintain to any other node outside of its cluster are called *intercluster* links. This work adapts these concepts to preserve domain awareness within *SkipMon*.

Gossiping enables messages, with a probabilistic guarantee, to reach a subset of nodes in a network without flooding. The probability of a message reaching all nodes within one cluster, that is, of every node in a cluster having a directed path to every other node within that same cluster, is given by [77]

$$p_n = \exp(-e^{-\beta}) \tag{1}$$

where n is the total number of nodes in the cluster, e is the Euler constant, and β is a constant. By fixing p_n to a desired value and solving for β, it is later possible to determine the number of required *intra-cluster* links k that each node in the cluster needs, for disseminating information efficiently, by utilizing

$$k = \log(n) + \beta. \tag{2}$$

SkipMon is also concerned with the preservation of domain awareness. This implies that not all nodes have the ability to contact or communicate with every other node outside of its domain. This is the same as restricting the number of *intercluster* links that exist between node clusters. If we consider each domain as a cluster, the number of *intercluster* links f required to guarantee a probability p_m of having all clusters (or localities) m connected with a path is defined as

$$f = \log(m) + \gamma. \tag{3}$$

Once again, the constant γ can be calculated by fixing p_m and solving for c in $p_m = \exp(-e^{-\gamma})$.

Alert Correlation

In the following, we provide insights of the alert correlation in *SkipMon*. For this, we first discuss the construction of alert messages, and subsequently present our similarity-based correlation mechanism.

Alert Messages

The alert messages produced by local IDSs may contain a lot and possibly redundant information for the purposes of a CIDS. In fact, this issue has been identified and resulted into extensive research into aggregation and correlation of alert data [138].

To cope with this, two important decisions are taken, in the context of representing the alert messages. First, we argue that only a small fraction of the alert messages' data is required for other sensors to be able to discover similarities. Bearing this in mind, the system can utilize a number of important features for representing alerts, e.g., the IP addresses of attackers, as well as the source and destination port numbers of an attack. Note that similar decisions are also taken in the majority of distributed CIDSs in the related work [172].

To handle and exchange alerts in a compact and privacy-preserving manner, *SkipMon* makes use of bloom filters [160]. Bloom filters are a probabilistic data structure that represents elements in a set and provides an efficient mechanism to check whether a particular element is part of the set or not. Bloom filters can handle a very large amount of data in an efficient manner. In addition, bloom filters are capable of preserving the privacy of alert data as no information can be leaked

out. In fact, they only possess the ability to check whether a certain element is part of the set or not. Therefore, organizations can use CIDSs that support the distribution of messages via bloom filters without revealing sensitive information. Moreover, bloom filters do not produce false negatives and the false positives ratio can be adjusted with the following equation [113]:

$$P_{fp} = (1 - (1 - \frac{1}{m})^{kn})^k, \tag{4}$$

where m is the number of bits in the bloom filter, k the number of hash functions that are utilized, and n the number of elements in the bloom filter. The aforementioned properties, and especially Equation 4, are important as they depict the applicability of bloom filters in the context of a CIDS.

SkipMon makes use of the bloom filters in the following way. Each sensor produces alerts from which specific features, e.g., the (adversaries) IP addresses are extracted, and subsequently added into a bloom filter. Afterwards, each sensor will utilize the available alert dissemination techniques (cf. the previous subsection) and send their bloom filters to other nodes.

Figure 39: Messages in *SkipMon*.

Overall, messages in *SkipMon* contain four different fields, as shown in Figure 39. The first field, *bloom filter*, contains, as the name implies, the actual bloom filter. In addition, the *sender node name* as well as the *message identifier* (i.e., a hash value) are added. Both of these fields are utilized for minimizing redundant messages and, thus, reducing the overall overhead when disseminating messages. Lastly, the *domain awareness value* (L) is an integer that defines the depth of *SkipNet* sub-domains that the message can reach. For example, a zero value ($L = 0$) indicates that domain awareness is disabled and the message can be disseminated to any sub-domain. A value of one ($L = 1$), however, would indicate that messages can be disseminated to a sub-domain if and only if the first field of the DNS name of two nodes is the same. An example, for three different L values, is also given in Figure 40.

Similarity Correlation

Alert correlation takes place when a node receives a message and determines whether the alert data received is relevant for the recipient node or not. The goal of correlating alerts is to provide a mechanism for connecting nodes that experience similar traffic patterns. Regard-

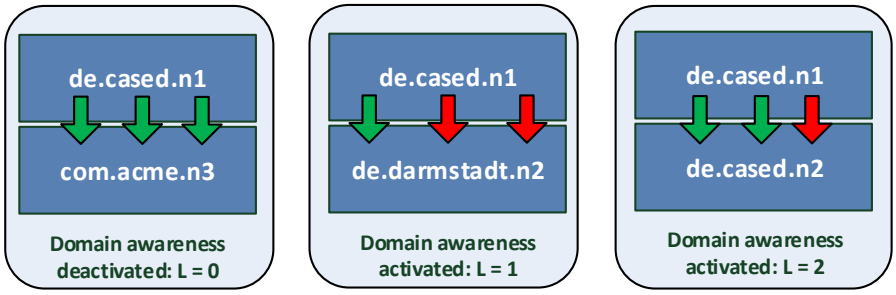


Figure 40: Domain awareness example in *SkipMon*.

less of the utilized alert dissemination technique, nodes receive messages from other nodes and compute their similarity value. For this, *SkipMon* exploits the inherent properties of the bloom filters and particularly their ability to perform logical operations such as the bitwise AND ($\wedge$) and the bitwise OR ($\vee$). To be able to do so, all bloom filters must have the same size and utilize the same hash functions. We define the similarity $S_{a,b}$ of two nodes n_a and n_a as

$$S_{a,b} = \frac{bf_a \wedge bf_b}{bf_a \vee bf_b}. \tag{5}$$

Each node is represented by the set of bits found in their bloom filters. The similarity correlation of two nodes is calculated by dividing the bitwise AND over the bitwise OR of their set of bits.

Equation 5, is essentially inspired from the Tanimoto similarity (similar to the Jaccard similarity). In the context of collaborative intrusion detection such an approach is very efficient for detecting similarities with respect to the overall alert data of a monitor. Therefore, it can be utilized for the, out of the box, detection of DDoSs attacks or for the identification of malware propagation.

After calculating the similarity value, nodes will make use of a threshold value t to determine whether S is similar enough or not. As it is discussed in the next section, the threshold creates a leverage in the number of proposed communities of sensors; when t is low, for example, a large number of sensors are found to be similar and therefore grouped together.

The problem of finding an optimal threshold value (golden standard) heavily depends on the network that is to be monitored. A large CIDS that consists of monitoring sensors that are exposed to attacks from the Internet is expected to have different properties from an internal CIDS that is focusing on the monitoring of a large internal corporate network and so forth. In general, an approach to deal with the optimal threshold problem can be to estimate the similarity distribution of random data. That is, inject into a large amount of bloom filters random data (bits) and calculate the similarity (S). With regard to the discovered similarity distribution the threshold (t) can be adjusted accordingly.

Community Formation

After the successful dissemination and correlation of the alert data, each sensor creates a matrix with its local knowledge of other sensors. Based on this knowledge and along with the utilized threshold, sensors can identify others and form a community with them to, afterwards, exchange more fine-grained alert data. An example of such a matrix is shown in Table 11. In the case where the threshold $t = 0.8$, node 3 (n_3) would only create a community with node 4 (n_4). Details on the exchange of alert data after the community formation are, however, out of the scope of this chapter.

Node	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$	$S_{3,5}$
n_3	0.5	0.7	**1**	**0.9**	0.4

Table 11: Example of similarity scores for node n_3.

IMPLEMENTATION

The prototype of *SkipMon* [48] is written in C++, it contains more than 6500 lines of code, and it is distributed under the GNU Lesser General Public License (LGPL) v.3. Figure 41 depicts an overview of the architecture of the implementation. It provides a detailed view on how different modules of *SkipMon* are connected. In the following, we briefly discuss each of them, i.e., the *Control Module* the *Node Management*, as well as the *SkipNet/SkipMon* sub-modules.

The *Control Module* is responsible for managing multiple *Node Management* instances, monitor their status, as well as (for the purpose of the evaluation) injecting alert data to the nodes.

The *Node Management* is responsible for reporting the status of *SkipMon* nodes (to the Control Module), for connecting sub-modules of the system and for providing an interface for exchanging routing information.

The first sub-modules that are started from the Node Management are the *SkipNet* nodes (implemented with respect to the details given in [67]). *SkipNet* nodes form an overlay that is used as a backbone for all the further operations that are done by the *SkipMon* sub-modules. The latter, store alert data into bloom filters, share them in the network, and correlate information received from other nodes as discussed in the previous section.

EVALUATION

This section provides insights and a discussion of the results gathered from the evaluation of *SkipMon*. The main research question lies

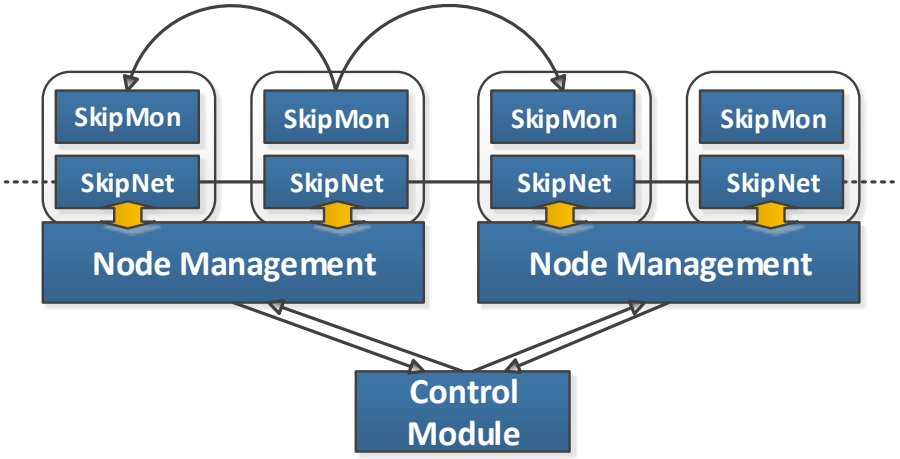


Figure 41: *SkipMon* implementation overview.

on the measurement of the performance of the proposed system in terms of its accuracy (in the context of finding sensors that experience similar traffic patterns) compared to a centralized system with full knowledge of the alert data of all participating sensors.

First, a description of the utilized dataset is given along with information regarding the evaluation setup. Afterwards, the results are discussed by studying the accuracy of the alert correlation technique and the domain awareness properties of *SkipMon*.

Dataset Description

For the purposes of the evaluation a dataset provided from DShield [165] will be used. The DShield project collects alert data that is sent in by volunteers, e.g., IDSs and firewalls, from all over the world. In more details, the utilized data, from a 24 hours period, consists of 7,841,775 alerts from 232,379 unique attackers, reported by 138,192 monitoring sensors.

Log ID	Malicious IP Address	Source Port	Target Port	Protocol	Sensor Hash
4616...	116.211.000.232	50978	8080	6	8078...
4502...	094.247.231.216	40370	5900	6	C58A...
4503...	087.118.541.555	3487	445	6	FCBE...

Table 12: DShield dataset example

Table 12 shows an excerpt of the data. The entry for each alert event is organized as follows:

- Log ID: A unique ID for each alert (truncated in Table 12).
- Source Port: The port that was used for the malicious activity.

- Target Port: The port that was targeted during the malicious activity.
- Protocol: The protocol number of the generated alert.
- Sensor Hash: A unique ID (i.e., a 160 bit hash) that serves as a pseudonym for each monitoring sensor providing data (truncated in Table 12).
- Time stamp: The exact time stamp (date and time) in which an alert occurred (excluded from table Table 12 due to space constrains).

The dataset was pre-processed, that is, the alerts were sorted with respect to the reporting sensor and stripped of any information other than the IP address (cf. Section 9.2.4.1). In addition, since IP addresses can occur multiple times per node, e.g. when (port)scanning multiple ports of the same sensor, duplicate IPs (targeting the same sensor) have been removed. Finally, only the data of the top contributing monitoring sensors was taken into account for the evaluation; sensors that provided less than 10 alerts were excluded from the evaluation.

Evaluation Setup

The purpose of the evaluation is to assess the accuracy of *SkipMon*'s mechanism of detecting similar sensors, to compare the different dissemination mechanisms, and lastly examine how the domain awareness property influences the accuracy of detection of similar sensors. For this, we discuss the results of 50 repetition runs, with 100 monitoring sensors that each of them contains a maximum of 1000 alerts in their bloom filters. The respective plots include the min/max values of the number of proposed communities for each threshold value.

For measuring the accuracy of detecting similar sensors a metric called *number of proposed communities* is utilized. This, as the name implies, refers to the number of (correctly) proposed communities of sensors and it is examined with respect to various similarity thresholds (by utilizing the Equation 5). Hence, to assess the accuracy of *SkipMon*, it is compared to a centralized system that possesses global knowledge of all the alert data of the participating sensors. For a more detailed examination of the differences of a centralized system with *SkipMon* the false positive and false negative metrics are used, that are defined as follows. *False positives* refer to the communities of sensors that were proposed in our distributed system, but were omitted in the centralized system. Similarly, *false negatives* represent the communities of sensors that have been proposed by the centralized system, but were not detected by *SkipMon*. Moreover, for measuring the communicational overhead, the total number of exchanged messages is examined.

Results

In the following a detailed discussion of the results is given with regard to the accuracy of the alert correlation and the domain awareness property. The main research question that is to be answered is how close, in the task of clustering monitoring nodes with similar traffic patterns, is *SkipMon* to a centralized system with full knowledge. In addition, the communication overhead trade off (as a result of utilizing a fully distributed system) is examined. Lastly, this section touches the topic of domain awareness and the influence that it has in the generation of communities.

Accuracy of alert correlation

For disseminating alerts in *SkipMon*, the flooding and gossiping mechanisms, as described in Section 9.2.3, are utilized. More specifically in the case of gossiping we make use of Equations 2 and 3 by setting the probability $p_n = 0.9$[1].

Figure 42 presents the results of flooding and Figure 43 the results of gossiping respectively. As one can observe the accuracy for both techniques is close to the centralized system. Moreover, as expected, the numbers of proposed communities in all cases significantly decrease when the threshold is increasing. Lastly, Table 13 depicts an overview of the communicational overhead, by counting the total number of messages that are exchanged in each of the three cases. The centralized system requires a lower number of messages but this metric does not take into account the computational overhead for the central component, or the need for scalability. In addition, as expected flooding generates significantly more messages than gossiping.

CIDS	Mean number of messages	Min	Max
Centralized System	452	452	452
SkipMon (flooding)	1812	445	4793
SkipMon (gossiping)	1346	290	2030

Table 13: Communication overhead comparison

For a more detailed look on the dissemination mechanisms in *SkipMon* we examine the false positive and false negative metrics. Figure 44 presents the results of false positives and negatives when flooding and Figure 45 when using gossiping. On the one hand, the amount of false negatives in the case of flooding can be attributed to information loss in the system, e.g., by dropped messages. On the other hand, the false negatives when gossiping occur due to the fact that not all

1 For further experiments with different k and p_n values of Equations 2 and 3 see Apendix C.

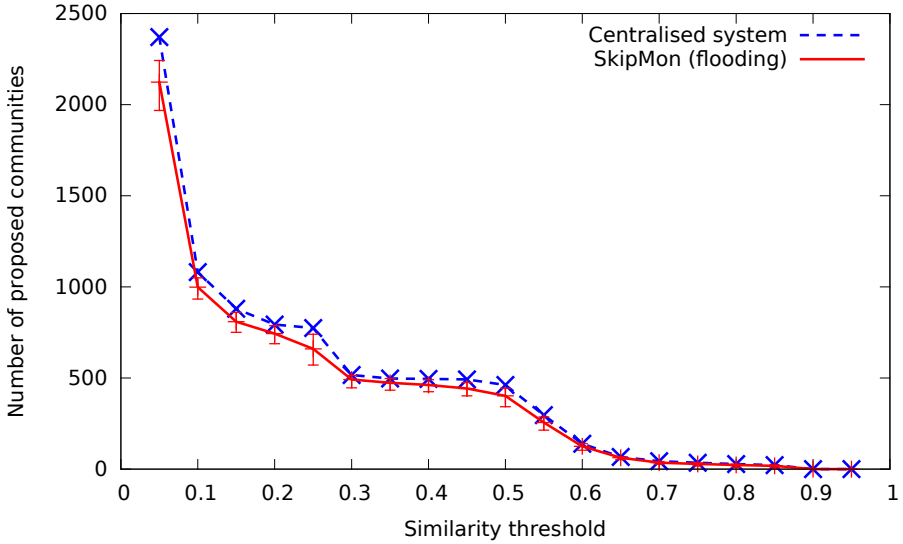


Figure 42: Proposed communities by *SkipMon* (with flooding) compared to a centralized system.

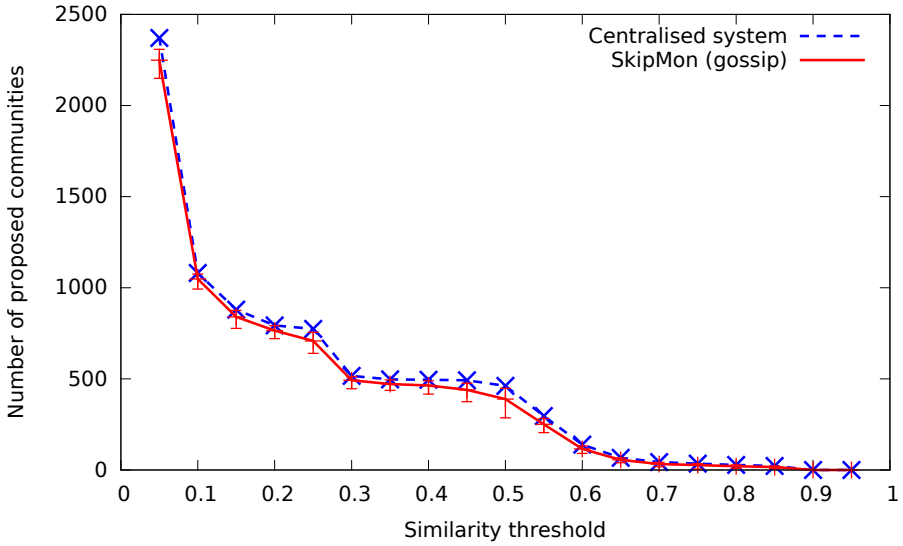


Figure 43: Proposed communities by *SkipMon* (with gossiping) compared to a centralized system.

nodes are communicating with each other. Moreover, with the number of total events decreasing, i.e., higher threshold, the number of false negatives also decreases. Finally, it is interesting to note that the total number of false positives is, in all cases, significantly low, due to the bloom filter utilization in the computation of the similarity.

Strict domain awareness

To evaluate the domain awareness properties of *SkipMon* four different domains with different DNS suffixes were created, resulting in different name ID prefixes. As the utilized dataset itself does not pro-

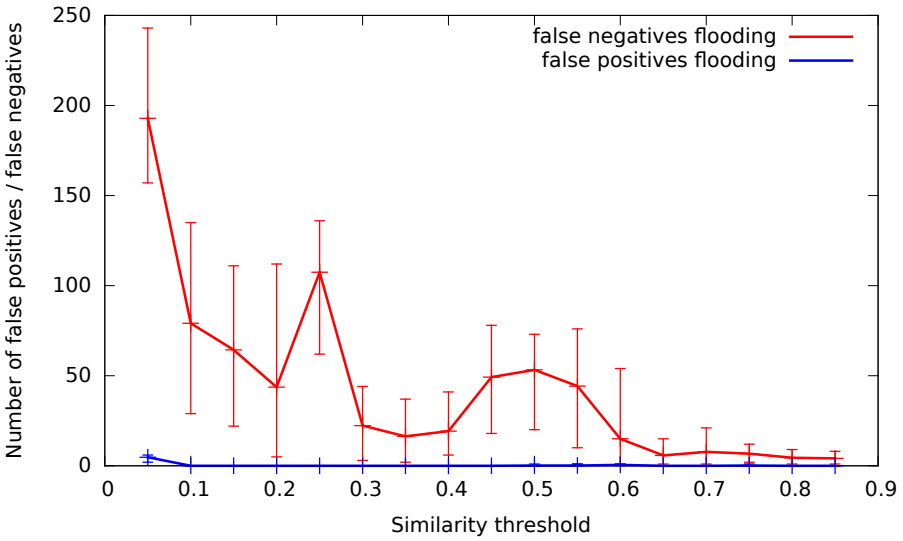


Figure 44: False positives and false negatives (flooding).

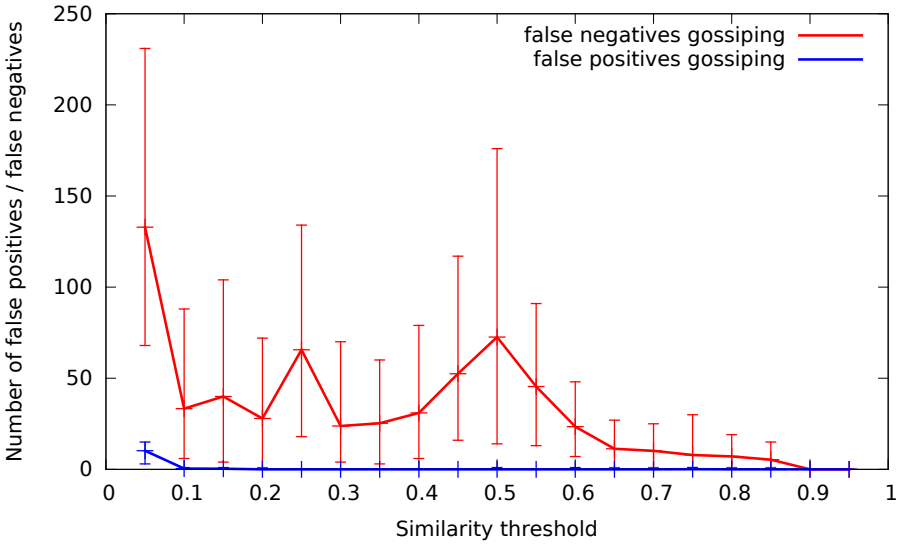


Figure 45: False positives and false negatives (gossiping).

vide any information regarding domains, the DShield IDSs' alerts are assigned to the monitoring sensors randomly. With respect to the domain awareness metric this reflects to $L = 1$ for all four domains, and the dissemination mechanism in this case is partial flooding (cf. Section 9.2.3). The results of the experiments are depicted in Figure 46. As expected, this configuration results to a much higher error rate compared to the centralized system, which does not follow any domain awareness constraints. Nevertheless, in a real world scenario, it is expected to have higher similarity in the alerts between nodes of the same domain and thus a higher number of proposed intra-domain communities between those nodes. Therefore, we argue that these results can be seen as the worst case scenario due to the enforced randomness in the alert creation level.

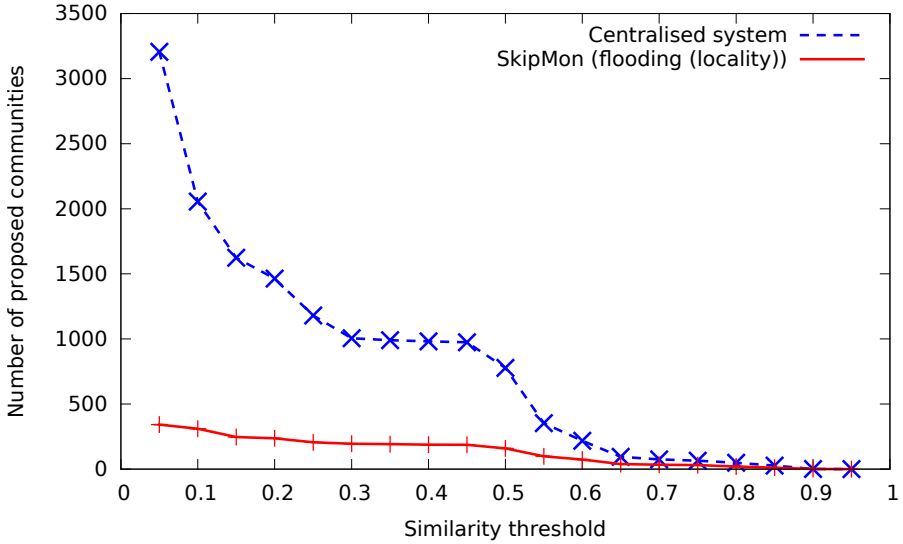


Figure 46: Strict domain awareness in SkipMon.

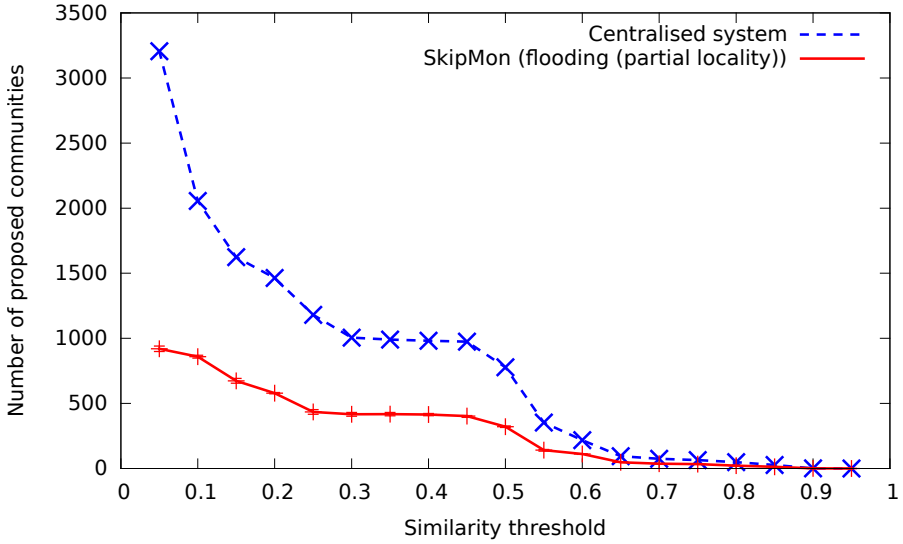


Figure 47: Partial domain awareness in SkipMon.

Partial domain awareness

In order to observe a case of a possible real world scenario the case of partial domain awareness is examined. That is, we examine how the system behaves when three of the domains keep the restrictions of internal dissemination (i.e., $L = 1$) and one of them is able to share its alerts with all nodes (i.e., $L = 0$). This scenario is depicted in Figure 47. As seen in the plot, the information shared publicly by a quarter of the nodes, enables *SkipMon* to find about twice as much communities among the nodes. Again, due to the smaller amount of messages flooded, the results of the nodes are denser over the runs.

SUMMARY

This chapter presented a novel distributed CIDS approach called *SkipMon*. The proposed system extends the state of the art in collaborative intrusion detection in two main aspects:

- The proposed *similarity-based correlation* mechanism can effectively correlate large amounts of alert data while preserving their privacy.
- *Domain awareness*, the ability to dynamically constrain alert dissemination with respect to security policies, is proposed, realized and implemented via the adaption of the SkipNet P2P overlay.

In addition, the proposed approach complements the CIDS area by providing the first publicly available distributed CIDS. Moreover, the adaption of the gossiping approach from Kermarrec et al., as seen in Section 9.2.3, contributes in the area of data dissemination by offering a gossiping mechanism for SkipNet that takes into account the domain awareness property of the overlay. Lastly, the given implementation is one of the first practical realizations of the SkipNet P2P overlay.

10 PROBE-RESPONSE ATTACKS

The previous chapters, in this part of the thesis, offered contributions in the design and the creation of CIDSs with respect to the building blocks presented in Chapter 4. This chapter deals with a specific class of attacks, namely *Probe Response Attacks (PRAs)*, which can be utilized for the detection of the monitoring sensors of a CIDS. In particular, a number of significant improvements are proposed for such attacks as well as for their mitigation. The aforementioned advances are realized with the development of a framework that enables many PRA-related actions. The remainder of this chapter is structured as follows. Section 10.1 provides an introduction and background knowledge with regard to PRAs as an extension of Chapter 3.3. Section 10.2 introduces a framework for the development of PRAs as well as the contributions of the thesis in the areas of attack mitigation and attack improvement. Afterwards, Section 10.3 provides insights from the evaluation both in terms of a simulation and real world attacks. Section 10.4 concludes this chapter. Finally, Figure 48 depicts the overview of the chapter with regard to the overall thesis structure.

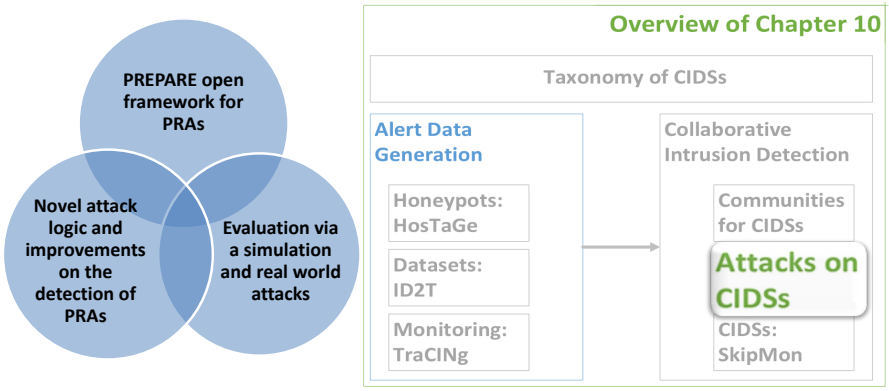


Figure 48: Overview of the Chapter and key contributions.

INTRODUCTION

Over the last years a number of CIDSs that adopt the role of a cyber-incident monitor arose, e.g., DShield [165] and TraCINg (cf. Chapter 6). A cyber-incident monitor can provide valuable insights by visualizing and correlating data from a large number of sensors. These systems are of high significance for a multitude of reasons; first, they are important for the scientific community, e.g., for studying attacks, experimenting with real-world attack data, creating statistics, etc. Second, they can be utilized for the detection and containment of malware propagation. For instance, DShield aided in the early detection of the Code-Red worm [115].

For every CIDS it is essential that the network position of its sensors, i.e., its IP address, is not revealed [172]. This is important for a multitude of reasons. First, an adversary with such knowledge might attempt to take down sensors, e.g., via conducting a DDoS attack. Furthermore, malware can utilize such knowledge to evade sensors and thus remain hidden for a longer period of time.

Probe Response Attacks (PRAs) are a specialized class of attacks against CIDSs that aim on detecting the network position of collaborative sensors, i.e., their IP addresses. PRAs take advantage of the need for publicly accessible alert data generated by CIDSs. In particular, they make use of the output given by a CIDS as a feedback loop towards learning information regarding the CIDS sensors.

As a whole CIDSs can be classified, based on their network architecture, into centralized, hierarchical and distributed [172]. In this chapter, the focus lies on CIDSs that publish their results publicly over the Internet. Even though most of existing systems in this category exhibit a centralized architecture, e.g., [165, 171], the applicability of the attacks discussed in this chapter is architecture agnostic; the sole requirement is access to the alerts generated by the CIDS.

PRAs were introduced by Lincoln et al. [91] and were further improved by several researchers, e.g., [13, 143]. An example of such an attack is given in Figure 49. The attack usually involves several steps, that can be summarized in the following. The adversary begins a PRA by dividing the whole IPv4 address space into equal groups (for the sake of simplicity, Figure 49 assumes two groups). Each group is assigned with special crafted watermarks, so-called *markers*; in the current example every three hosts are assigned with the same probe. Afterwards, the adversary launches the attack by sending a very large number of probes in the respective address space. The driving idea behind such a *divide and conquer* attack is that the markers can be subsequently utilized for examining the output of the CIDS and determining whether it contains signs of the markers or not. In this context, and with respect to the received output from the CIDS, the

attacker can reduce the probed IP space and repeat the probing steps until the addresses of the sensors are revealed.

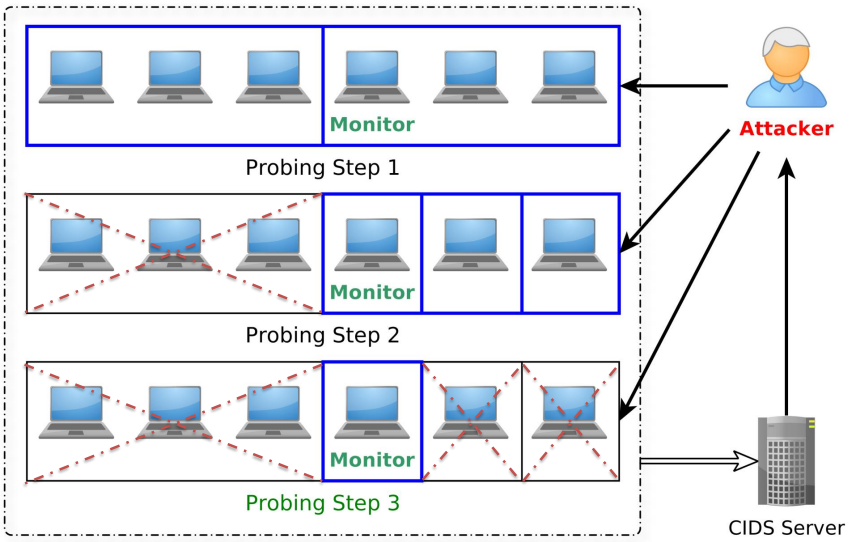


Figure 49: Probe Response Attack (PRA) example [175].

This chapter proposes several improvements on both the PRAs and also their detection and mitigation. In more details, an open-source framework, called *PREPARE*, is introduced that can be practically utilized for performing PRAs as well as for studying mitigation techniques. Moreover, a number of novel mechanisms for improving PRAs and also for defending against them is proposed. The framework and the aforementioned improvements are evaluated in a simulation environment and by deploying real-world attacks on two different CIDSs. The results (cf. Section 10.3) suggest that the proposed techniques significantly improve the efficiency of PRAs. In addition, the detection and mitigation mechanisms can be practically realized.

PREPARE

This section firstly discusses *PREPARE* a framework for the development and execution of PRAs, along with its structure and properties. Afterwards, insights are given with regard to the implemented attack improvements as well as the proposed novel attack mitigation techniques.

System Overview

Figure 50, depicts an overview of the framework's architecture. PREPARE is written in Python and C and can be split into three main blocks,

the *User Interface (UI)*, the *PRA logic*, and the *Wrapper* (that includes the scanning mechanisms).

The UI of PREPARE is a typical console-based interface that provides the user with all the basic commands for customizing the parameters of a PRA. The *PRA logic* implements the proposed attack methodology based on a certain logic flow that is described in the following section. The *Wrapper* contains a modified version of the core of the ZMap scanner [42]. In more details, ZMap was extended by the addition of several new modules that are responsible for packet generation, response interpretation, and for handling the output. As one can observe from Figure 50, the *PRA logic* makes use of ZMap by first providing, as input, the optimal configuration (e.g., the specific marker strategy, the scan rate, etc.) and afterwards receiving and analyzing the scan results. After the successful completion of an attack the framework generates a CSV file, that contains all the information regarding the identified sensors.

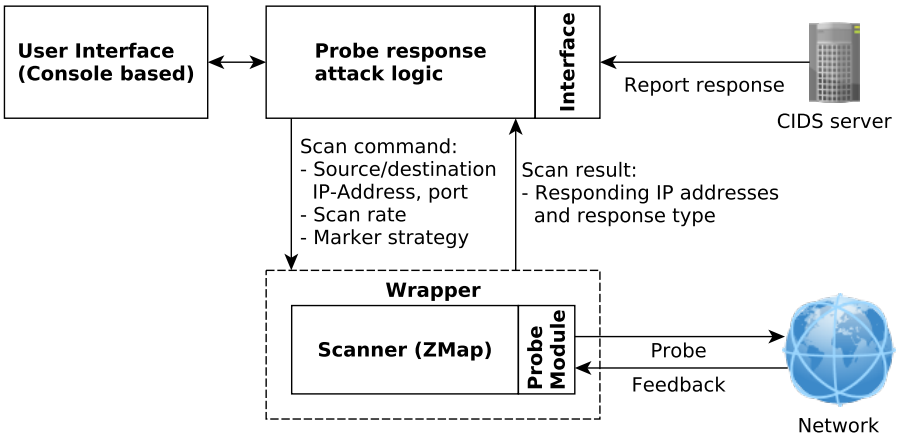


Figure 50: PREPARE attack framework's high-level overview.

Improving PRAs

The basic principle of PRAs is to correlate attack events, from the output of a CIDS, to probes by utilizing markers. To better understand how markers can be constructed, an example is given in the following that describes the utilization of destination ports as markers (via basic marker-encoding).

The idea of address encoding, enables the attacker to map target addresses of sensors into a port range. For instance, by encoding the first two bytes of a destination IP address range of 0.0.0.0 to 255.255.0.0 into a port range of 0 to 65535 (0 = 0.0.0.0, 1 = 0.1.0.0, etc.), an attacker is able to, later on, decode this address. Subsequently, this allows to reduce the address range to be scanned. Based on the last example, if only the port value 1 is received from the attack report of the CIDS, further scans can be limited to the subnet 0.1.0.0/16. Assum-

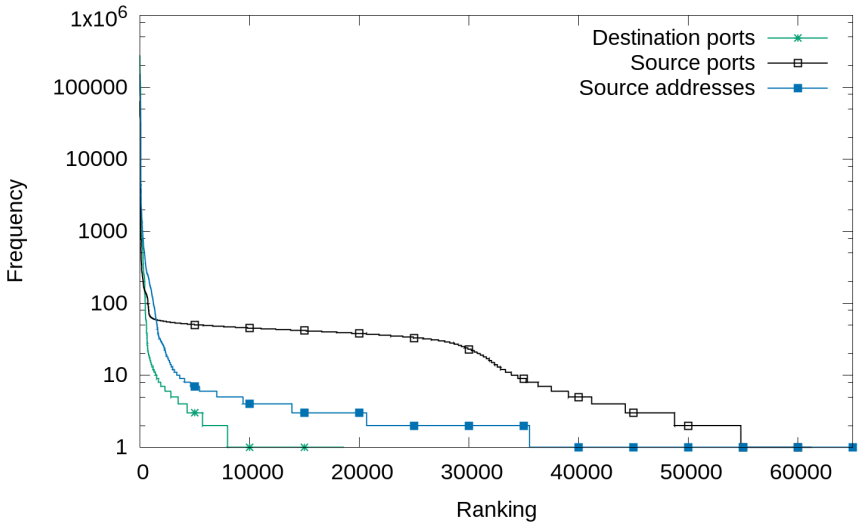


Figure 51: Distribution of possible markers in DShield.

ing that the source and/or target ports are shown in the report, the attacker is able to read and decode the port information and apply this encoding methodology without additional effort.

However, the aforementioned encoding logic (introduced by Bethencourt et al. [13]) does not take noise into account. *Noise* refers to ordinary attacks, which appear in the CIDS's output, that can be falsely interpreted (by the adversary) as part of a PRA. Hence, from the attackers' perspective, noise is important as it can introduce *false positives* and it can be seen as a two-dimensional problem. First, there is the case that a CIDS produces alerts in which the ports have high density. This degrades the effectiveness of a PRA as the number of noise-free ports (utilized as markers) is low, and thus can generate many false positives. Moreover, when the alerts include only a few ports, but the amount of the alerts is very high, this can also affect the bandwidth requirements of a PRA and the amount of re-probing required. Diverging from previous work a novel marker-encoding methodology is proposed that also takes into account noise.

First, the utilized marker *type* is not limited to a specific field but can be rather dynamic with respect to the specifics of the targeted CIDS. For instance, Figure 51 depicts the frequency distribution of possible probe markers, i.e., destination ports, source ports, and IP source addresses, in the context of the Dshield CIDS [165]. More specifically, the figure plots the frequency of the alert data gathered in a 12 hours period. From the set of all available ports, only a few are ever utilized, and also the IP addresses provide enough space for a marker. In more details, approximately $46,943$ destination, and $4,270$ source ports do not appear in the analysis, which provides enough flexibility to utilize them as markers. This also applies to the (source) IP addresses (in a magnitude of 10^9 available addresses) especially when taking

into account that an attacker can spoof IP addresses that have not been seen before. Thus, introducing a combination of probe types, effectively multiplies the amount of the available markers.

The approach proposed in this section, called Generic Marker Encoding Methodology (GMEM), combines all available marker *values* of a CIDS (e.g., source/destination ports, source IP addresses, etc.) and introduces a *checksum* along with the encoded marker. The latter offers a highly effective remedy for noise, as all markers need to pass an encoding phase before considered part of a PRA. The total amount of available marker bits M_{bits} can be calculated by multiplying the sizes of all markers m as $M_{total} = \prod m_i$ and deriving $M_{bits} = \log_2(M_{total})$. For instance, the marker types when utilizing the source and destination ports[1], results in $M_{total} = 65535 * 65535 = 4.294.836.225$ which gives $M_{bits} = 32$.

Figure 52, depicts the overall activity flow for a PRA that utilizes GMEM, which is split in four logical steps, namely: *Pre-selection*, *Encoding*, *Probing* and *Decoding*.

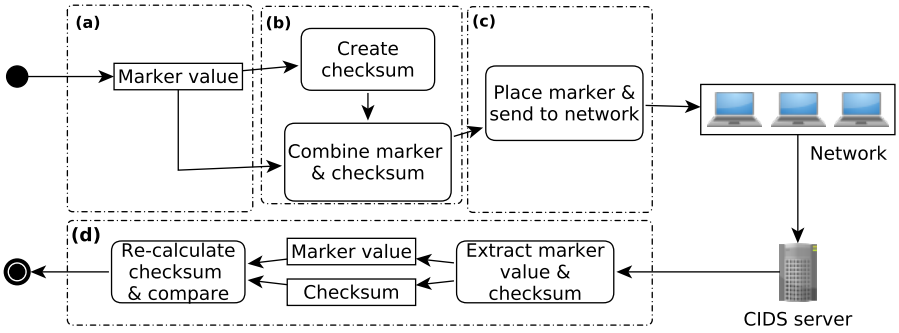


Figure 52: GMEM flow overview example.

PRE-SELECTION In the first step all available marker types are concatenated in a specific order. This generates a specific marker *pattern* that is afterwards used for inserting the marker value and the checksum. As an example the following marker types can be used:

- A: Destination port (16 bits)
- B: Source IP address (32 bits)
- C: Source port (16 bits)

The resulting marker pattern P can be presented as [AAAA][BBBBBBBB][CCCC], where every upper case letter represents four bits. Note, that intermixing individual bits is also allowed as long as the pattern maintains its structure throughout all the steps of GMEM.

1 65635 is the total number of available TCP and UDP ports.

ENCODING In this phase, the actual marker value, e.g., the (candidate) IP address of the target sensor, is placed in the marker pattern. The IP address in this case can be represented as DDDDDDDD, and can be placed in the beginning of the pattern, transforming P to [DDDD][DDDDBBBB][CCCC]. After encoding the marker value, a marker checksum is calculated over the previously defined marker value. This checksum in turn gets placed at the end of the marker after setting all unused bits to 0. With respect to the encoding example, the marker pattern P would become [DDDD][DDDD0000][0000] before generating the checksum $C = checksum(P) = SSSSSSSS$ and appending it to the end of the marker value, which becomes marker $m = P||C = [DDDD][DDDDSSSS][SSSS]$. Note that this simple concatenation can be exchanged with more sophisticated combinations of marker values and checksums as long as the same procedure is reversely applied in the decoding part.

PROBING When the first two steps are completed the *probing* phase can begin. Here, the generated marker m is placed into the destination field of the network packets to be sent.

DECODING Lastly, by reading the feedback of the targeted CIDS the decoding phase takes place. In this step, individual markers get extracted and ordered. The system calculates the checksum[2] over the marker value and compares it to the extracted checksum. In the case of a match the response is marked as accepted and can be further utilized to create subgroups and finally identify sensor nodes. Responses that fail the check are assumed as noise and hence are ignored.

GMEM introduces a trade-off between noise avoidance (filtering out more noise by utilizing more checksum bits but use less bits to build markers) and the amount of marker values (more bits for markers but less bits for checksums). As an example, two marker types A and B both providing four bits, could be used to create a total of 256 markers. Using all eight bits for marker encoding without any checksum would, however, lead to a high number of false positives. Alternatively by using six bits for address encoding (four bits from A and two bits from B) and two bits for a checksum (two bits from B), the total amount of encode-able markers would be reduced to 64. As the PRA defender cannot know which bits were taken for address encoding or which checksum algorithm was used, noise has to be introduced for the whole target range of 256 addresses. This reduces the probability of successful noise integration, that is the probability that an introduced value matches a correct encoded attacker value. It should be noted that the filtering effectiveness increases with the

2 Note that PREPARE is currently utilizing the hashing algorithm Fletcher32 for its efficiency [158], but this can be modified if needed by the user.

amount of attack rounds because the introduced noise would have to match the probed value in every new iteration, until it introduces a false positive in the final probing. In Section 10.3.2.1, a comprehensive study of the aforementioned trade-off is given to better understand it and to derive the most effective parameters for a PRA.

Attack Detection and Mitigation

This section discusses two novel mechanisms for defending against PRAs. The first one is focusing on the detection of such an attack, and the second one on reducing the effects of a PRA dynamically (i.e., upon detection).

PRA detection

The first step to cope with PRAs is to detect their presence in a CIDS. This chapter proposes a statistical anomaly detection technique that is based on the following assumptions. First, in a generic CIDS scenario the adversary has no knowledge of either the IP addresses of the sensors nor the exact amount of them. In practice, this is realized by the need for a large-scale probing, e.g., the whole IPv4 address space. As a consequence of the first assumption, it can also be expected that a large amount of sensors will be triggered during a PRA. Therefore, the following statistical properties are expected during PRAs:

- In a certain time-window the amount of unique sensors generating alerts is significantly increased.
- The number of unique destination (and/or source) ports will also increase (assuming probes are sent out using port-based markers).
- The number of unique source IP addresses will also increase (assuming the utilization of spoofed addresses by the adversary).

Bearing the above in mind, the chapter proposes a simple, yet effective, metric to detect such attacks by utilizing the *ratio* of generated alerts in relationship to the number of actively reporting sensors. Let A be the set of all generated alerts, S be the set of all sensors, $S_t \subset S$ the set of reporting sensors within time-frame t, and $A_t \subset A$ the set of generated alerts within time-frame t. The ratio r_a is defined as:

$$r_a = \frac{|A_t|}{|S_t|}. \tag{6}$$

To better understand the applicability of such a metric, an emulated PRA scenario is given in which the respective values are calculated with data gathered from the DShield CIDS. In more details,

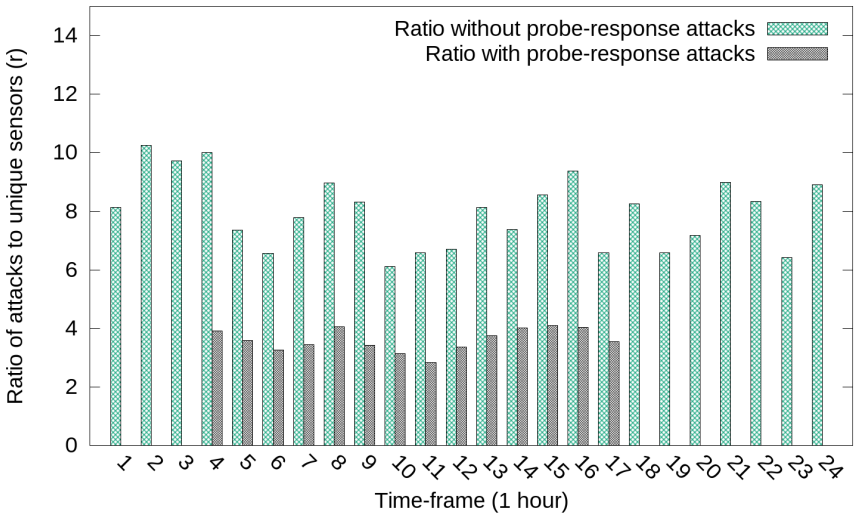


Figure 53: Ratio r_a utilization example for DShield data.

Figure 53 depicts the distribution of r_a for data gathered by DShield within a period of 24 hours. An attacker requires approximately 5 hours (with a 100Mbit/s network connection) to perform one probing step in the entire IPv4 range [42], probing approximately 90,000 sensor addresses per hour (assuming a total of 500,000 sensors). With respect to the aforementioned assumption, in the presence of a PRA the number of (unique) reporting sensors within a time-frame $|S_t|$ will increase significantly, while $|A_t|$ will only have a relatively small increase, therefore modifying r_a. In the presented period one can observe the sensors $|S| = 131,344$, the alerts $|A| = 10,934,768$, and an average unique sensor count (per hour) $\sum_t \frac{|S_t|}{24} = 55,000$. A PRA was emulated by introducing alarms in the time-frames between 4 and 17 (which enables three complete probing steps) in a 24 hour period. By assuming that the maximum probing rate is 90,000 and that sensors might already be present, the PRAs are injected according to a uniform distribution between 80,000 and 90,000. As it is depicted in Figure 53, it becomes evident that during an attack the ratio r_a decreases significantly.

Another technique for detecting the presence of PRAs is by studying the *frequency* of unique destination ports in a specific time-window. In contrast to source ports (that are usually chosen randomly), destination ports can be utilized as markers and thus their number is expected to increase during a PRA. In this case it is important to carefully decide which time window should be taken for studying the respective port frequency. Figure 54, shows the distribution of port frequency in DShield by setting a fixed start time and extending the window up to 24 hours. As one can observe, in the first half hour almost 93% of the ports are not utilized, while when the window is increased this percentage is decreased rapidly. This suggests

that large time-windows (e.g., more than two hours) might introduce many false positives.

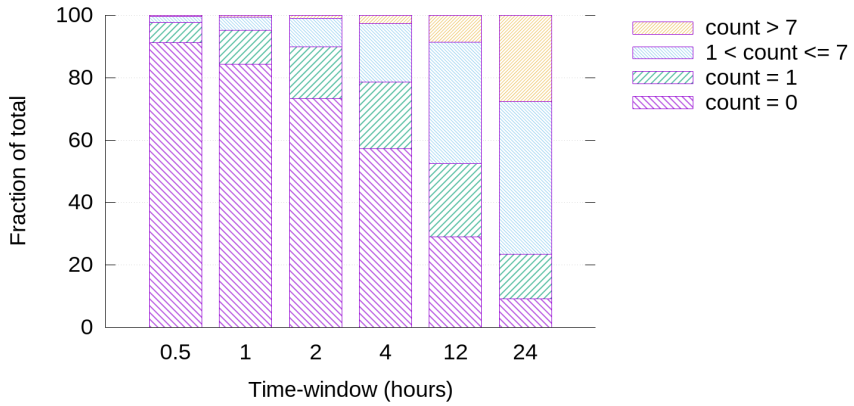


Figure 54: Destination port frequency for different time-windows in DShield.

Bearing this in mind, Figure 55 (with a similar setup as Figure 53) shows how the frequency metric evolves under the presence and absence of an emulated PRA. It can be seen that the difference between attacked and non-attacked states can be utilized as a threshold for the detection of PRAs. Section 10.3.2.2 shows how the frequency of the non-utilized ports can be used for the detection of a PRA.

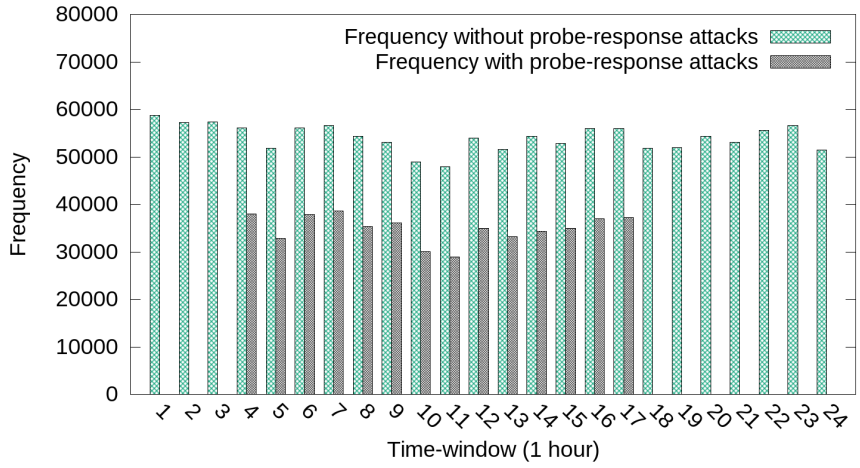


Figure 55: Destination port frequency in DShield.

Adaptive Reporting

Upon the successful detection of a PRA the CIDS can perform a number of actions that aim on the reduction of the results of the attack. Hence, the main goal is to reduce the number of identified sensors as much as possible. In this context, this section proposes the concept of adaptive

sampling, i.e., the CIDS will selectively publish a sample of the overall generated attacks whenever it detects the presence of a PRA.

Such a mechanism can make use of the aforementioned ratio and/or the destination port frequency metric to decide when the sampling should be activated. Furthermore, the intensity of the sampling can be dependable of the attack intensity, i.e., the more PRAs the less results are published by the CIDS. Section 10.3.2.2, describes two practicable variations of such an adaptive sampling approach. Furthermore, the efficiency of the two aforementioned detection techniques will be studied, combined with the adaptive reporting sampling, in more detail. As it will be shown, such an adaptive approach can efficiently reduce the effectiveness of a PRA. However, this comes with a trade-off as the published results of the CIDS will significantly reduced.

EVALUATION

This section discusses the results of the evaluation for both the attack and mitigation strategies that have been proposed in the previous sections. The evaluation is composed by an extensive simulation and experiments into two real-world cyber incident monitors.

Simulation Setup

In order to evaluate attacks and their proposed mitigation mechanisms, a simulation environment was setup. The simulations match the characteristics of DShield [165]. DShield is the largest and most well known cyber incident monitor, reporting thousands of potential attacks every day since ten years ago. Along with the DShield characteristics, the simulation also takes into consideration previous work in the area of Internet-wide scanning as the proposed methodology relies on scanning the entire range of IP addresses exposed on the Internet.

All the simulations use the following parameters. A set of approximately 288.4 million responsive IPv4 addresses is utilized, as identified in [98, 68]. Within all these responsive addresses, a total of 500 thousand monitors is set randomly. This is the same number of monitors that DShield is utilizing [165]. In addition, as network traffic does not always reach its destination, the simulation also takes into account a 2% packet *drop rate*. This particular drop rate has been observed in related work [68, 42]. Lastly, a low *bandwidth*, i.e., 56Mbit/s, is utilized so as to mimic the bandwidth available to many users (in offices and households).

Simulation Results

The simulation evaluation is split into two parts. The first part deals with the proposed improvements for PRAs while the second one with the suggested mitigation mechanisms.

Improving PRAs

First, the effectiveness of GMEM is studied along with the efficiency of such proposal in the presence of noise. In this context, noise was introduced by adding real-world data from DShield in a rate of 24 events per second (which corresponds to approximately two million attacks per day).

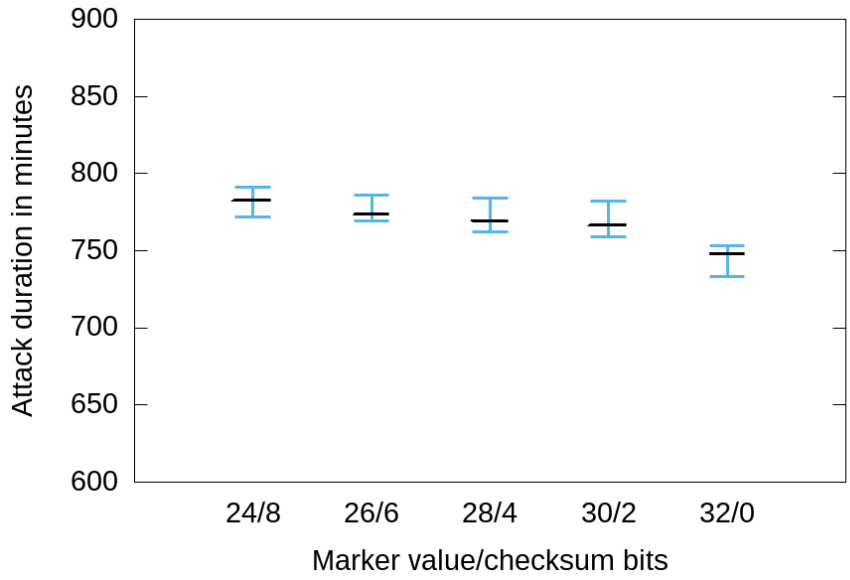


Figure 56: Attack duration with respect to marker values and checksum bits.

Figures 56 and 57, present the attack duration and the amount of required probes to perform the PRA for different marker values and checksum bits combinations respectively. The figures show that an increase on the bits of the marker's value effectively decreases both the attack duration and also the overall numbers of probes required. This can be explained due to the fact that the group size is becoming smaller (with increased marker values) which translates in a faster identification of empty or fully identified groups. Nevertheless, as it is shown in the following a trade-off exists between not utilizing checksum bits and the increase of false positives. In particular, this is the case in which noise is taken into account.

In more details, the false positives that are introduced by noise are examined, and how the proposed checksum mechanism can assist in their reduction. Figure 58, depicts the false positives when utilizing various marker values and checksum combinations. As it was ex-

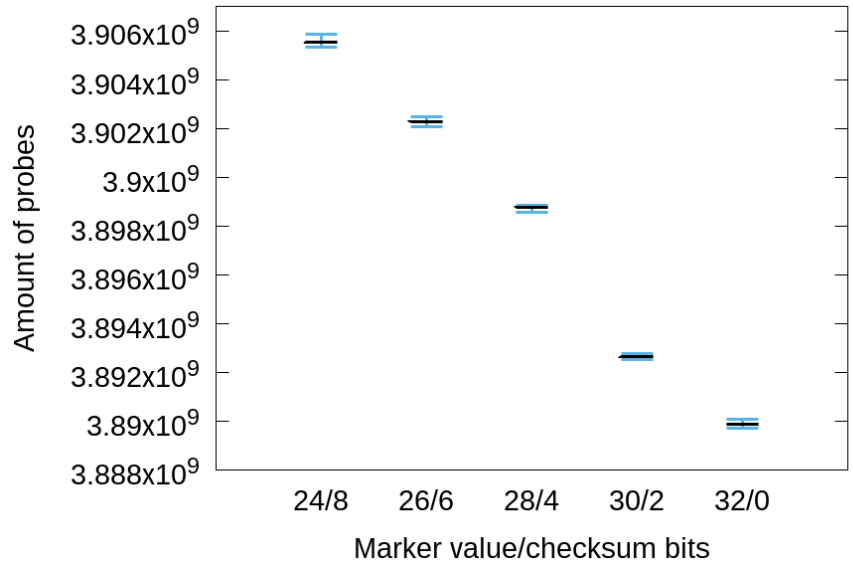


Figure 57: Amount of required probes with respect to marker values and checksum bit.

pected, the introduction of checksums decreases the amount of false positives in almost an exponential rate.

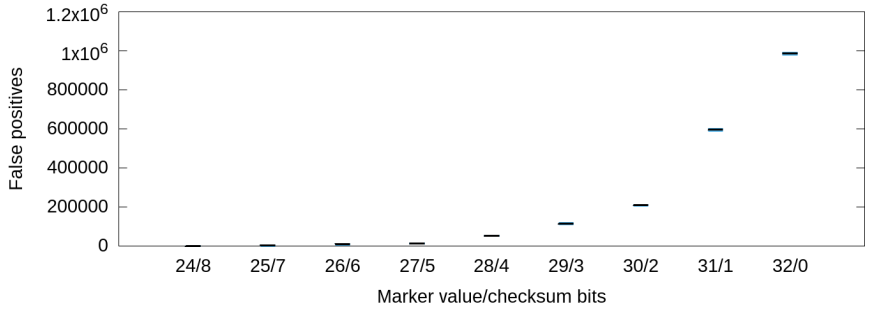


Figure 58: False positives for various encoding configurations.

Furthermore, Figure 59 compares our approach with the one proposed by Bethencourt et al [13]. The time required for the full enumeration of the sensors is plotted by utilizing the PREPARE framework, with a 24/8 marker value and checksum bits configuration and compare the results with the ones given in [13]. The results show that a significantly improved performance is achieved for detecting the complete fraction of CIDS sensors. In addition, PREPARE, with a bandwidth of 56Mbit/s, performs better even when comparing it to the fastest case of Bethencourt et al. (384Mbit/s).

Improving Mitigation

In the following, experiments are given with a focus on studying how well the proposed ratio-based mitigation and detection mechanism

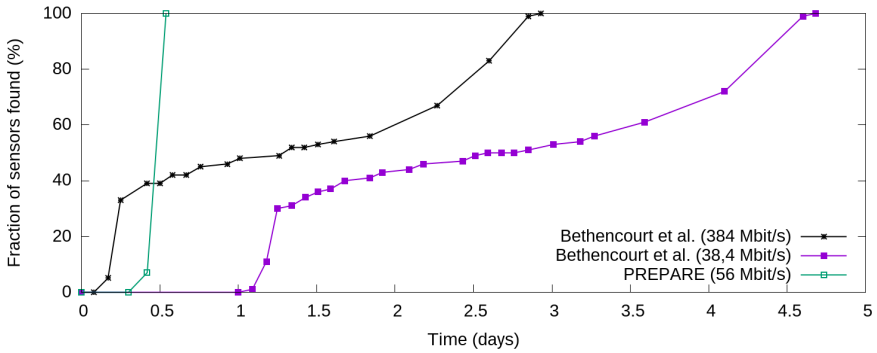


Figure 59: PRA comparison: time required for the complete enumeration of sensors.

perform. The driving idea here is to utilize the ratio-based detection to first detect a PRA and afterwards perform sampling to reduce the effects of the attack. Hence, a reduction of the detected sensors is expected, but a also reduction in the total number of events published from the CIDS as a result of the sampling process.

For this, the simulation framework is configured to perform PRA detection, and then adaptive sampling when the ratio drops below a certain threshold. Sampling in this case refers to the probability that an attack event is shown in the published results. The sampling is also adaptive in the sense that the lower the ratio is, the less the sampling is; in other words the sampling reacts to the intensity of the PRA.

DShield data was extensively analyzed to be able to select a ratio that, in the presence of PRAs, will not generate false positives. The conducted analysis showed that a threshold ratio that is between $(3, 4]$ will avoid false negatives and false positives. Hence, a threshold of $R_t = 3$ is utilized. With regard to the sampling, the formula

$$s_1 = \frac{R_m - 1}{R_t - 1} \tag{7}$$

is utilized, where R_m is the measured ratio and R_t the threshold ratio. As the minimum value for the ratio is one (every appearing monitor submits at minimum one event), the subtraction of one allows to a reach a theoretical sampling minimum value of zero.

Figure 60 depicts the development of the ratio of attacks to unique sensors, under the presence of a PRA. The ratio metric is checked every 60 seconds, i.e., one time-slot. The ratio (as already shown in Equation 6) is calculated by counting attacks and unique sensors for the whole time-window (one hour). The initial window state is set by loading one hour of DShield data without introducing any additional changes. In total, the attack duration was 671 minutes by sending a total of $3,555,452,622$ probes. The attack starts at time-window two and ends at time-window 11. As one can observe, there is a constant drop of the ratio starting with 2.7 and moving towards 1.5 until time-window 11. By utilizing sampling when the threshold detection is

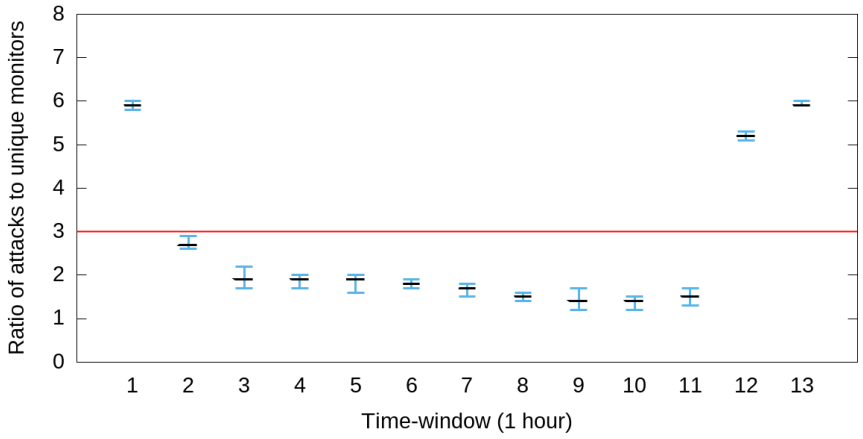


Figure 60: Attacks/Monitors ratio (r_a) development under a PRA.

active, the PRA resulted in the identification of only the 30.983% of the total sensors. However, it should be noted that this technique also results in a reduction of 62% in the total number of events reported by the CIDS as a result of the sampling.

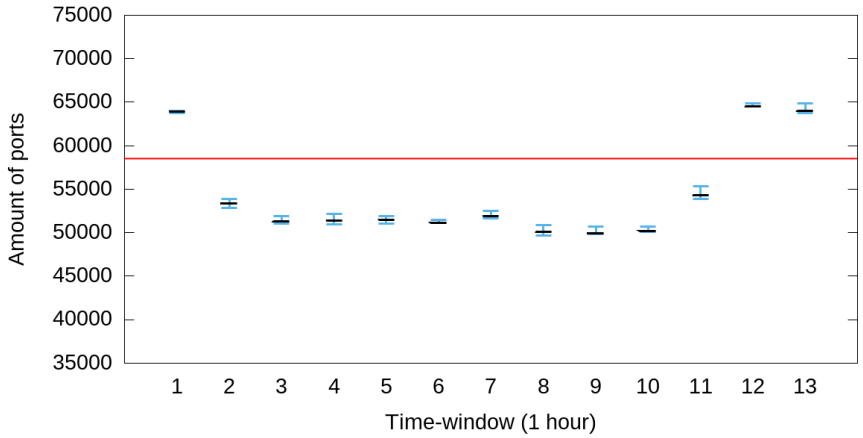


Figure 61: Development of non-attacked destination ports under a PRA.

Similarly, Figure 61 shows the development of non-utilized ports under a PRA. In this case, the sampling is done by utilizing the formula

$$s_2 = \frac{P_t}{P_a} \tag{8}$$

where P_t the threshold ratio and P_a is the amount of attacked ports. With this mitigation mechanism the PRA detected only the 26.816% of the sensors, but also resulted in a sampling reduction of 70% in the number of reported events.

Real-World Experiments

For a further evaluation of the proposals of this chapter, the PRA methodology proposed in this chapter was applied against two CIDSs: TraCINg and DShield.

TraCINg was tested with a bandwidth of 32Mbit/s, a marker value of 24 and a checksum of four. The overall attack duration was 1114 minutes sending a total of 3,621,468,528 probes. Overall, the 100% of the sensors have been identified, without introducing any false positives. The correctness of the results was confirmed both by manually re-probing the identified sensors and also based on our own ground truth knowledge of the network location of the monitoring sensors (as this CIDS is deployed by our university).

Top 10 Ports

by Reports		by Targets		by Sources	
Port	Reports	Port	Targets	Port	Sources
1337	2048379	161	48495	23	8788
22	546609	22	39364	80	7169
23	421783	1234	18564	445	6274
21	343886	23	5705	51413	6029
80	104923	3389	2026	53	4525
161	60277	3306	1971	443	1516
51413	31747	8080	1963	3389	1270
53	29291	9200	1832	25	1183
1234	23228	401	1779	22	1076
443	22701	1723	1735	3101	976

Figure 62: Top 10 Ports after the execution of a PRA as generated by DShield.

A PRA was also performed to DShield. For this, a marker value of 32 bits was utilized and no checksum. The checksum was excluded in this PRA as the attack report can be utilized after the attack to validate the results. Hence, all available marker bits can be used for probing. The utilized bandwidth in this case was 14,4Mbit/s to minimize the probability of abuse complaints. The duration of the PRA was 2071 minutes and resulted in the identification of 1932 sensors, geographically distributed all over the world. Similarly to case of *TraCINg* it was manually confirmed that the detected sensors did not include any false positives, by manually re-probing the sensors and examining the output of the CIDS. However, it is not possible to evaluate the case of false negatives in DShield, due to the lacking of ground truth knowledge. Note, that in the past DShield was claiming to utilize around 500,000 sensors, which does not correspond to the cur-

rent findings. Nevertheless, it can be that a number of sensors are not publicly reachable (e.g., they might be placed behind firewalls or monitoring local networks only).

Finally, Figure 62 is taken from the public output DShield after the PRA. As one can observe the utilized marker, i.e., port 1337, dominates the results and is considered the top attacked port. This also illustrates the easiness of not only performing a PRA but also tampering the results generated by a CIDS. For instance, a malicious entity could utilize such an attack to hide attacks manifested in a certain protocol/port.

SUMMARY

Probe Response Attacks (PRAs) can considerately reduce the benefits of CIDSs and in particular of cyber-incident monitors that publish their results publicly.

This chapter contributes in the area of PRAs in three distinctive ways:

- First, an open-source framework, the Probe REsPonse Attack fRamEwork (PREPARE), is proposed that enables the development, improvement, and execution of PRAs.
- Second, a novel attack logic, the Generic Marker Encoding Methodology (GMEM), is developed that significantly improves PRAs while taking into account the existence of noise in the CIDS's output.
- Lastly, a number of novel techniques are proposed that focus on the detection and mitigation of PRAs.

In addition, this chapter introduces one of the first studies that practically examine the applicability of PRAs in real-world scenarios. The evaluation results suggest that PRAs can be launched in a practical manner and severely reduce the advantages of a CIDS. Lastly, the interconnection of the chapter at hand with Chapter 6 provides with a holistic and promising environment for the further study of CIDSs.

Part IV

EPILOGUE

The last part of the thesis summarizes the aforesaid research contributions of this dissertation. In addition, a discussion of proposed future work is given for all the respective contributions.

11 CONCLUSION AND OUTLOOK

I don't write a book so that it will be the final word;
I write a book so that other books are possible,
not necessarily written by me.

— Michel Foucault

This thesis contributes in the area of collaborative intrusion detection, as presented in the last six chapters. In particular, the contributions have been categorized into two classes. First, chapters 5 to 7 emphasize on the alert data generation by introducing a novel honeypot, a cyber incident monitor and a toolkit for the generation of IDS datasets. Subsequently, chapters 8 to 10 contribute to the core areas of collaborative intrusion detection by proposing the concept of communities, a novel distributed CIDS and improvements on attacks for CIDSs. This final chapter concludes the dissertation by providing the reader with a summary of all contributions. In addition, the chapter presents an outlook of possible future work for selected areas of the two core parts of the dissertation, i.e., the alert data generation and the collaborative intrusion detection.

CONCLUSION

This dissertation contributes in the areas of collaborative intrusion detection and alert data generation for the evaluation of CIDSs and IDSs. First, Chapter 4 proposes a novel and detailed taxonomy for CIDSs and offers a comprehensive survey of the state of the art. Chapters 5, 6 and 7 deal with the topic of generating and handling real world and synthetic alert data for evaluation purposes. Afterwards, Chapters 8 and 9 deal with core areas of collaborative intrusion detection. In the following, the contributions are highlighted for each chapter of the two parts of the thesis.

Alert Data Generation

The second part of this thesis, and specifically Chapters 5, 6, 7, present several key contributions in the area of alert data generation in the context of collaborative intrusion detection.

- Chapter 5, introduces the idea of mobile honeypots. Particularly, *HosTaGe* is proposed, a honeypot that is able to run in mobile devices and emulate several protocols. The chapter formally describes the system, on the basis of EFSMs, and focuses on the detection of attacks with an emphasis on ICSs. *HosTaGe* is also the first honeypot to tackle the topic of multi-stage attack detection by performing similarity-based correlation on the identified attacks. Moreover, the system is emphasizing in its ability to complement existing security solutions by generating signatures of attacks for existing IDSs. The evaluation part of the chapter depicts the ability of the honeypot to detect attacks with similar or better accuracy than the state of the art. In addition, the evaluation section opens a discussion towards the topic of honeypot evasion. The initial results corroborate the hypothesis that *HosTaGe* can remain undetected. Finally, additional evaluation information with regard to *HosTaGe* is given in Appendix A.

- *TraCINg* a cyber incident monitor that also makes use of *HosTaGe* sensors is presented in Chapter 6. The system contributes to the state of the art by offering an open source platform for studying cyber-attacks and their effects. The chapter also offers a long term study of hopeypot worldwide deployment. This allows to study attack trends and also apply correlation algorithms for examining the connections between different attackers that target multiple monitoring sensors. Lastly, *TraCINg* can be utilized to experiment on the area of PRAs, as seen in Chapter 10.

- Chapter 7, serves as a first step towards the generation of synthetic, yet realistic, intrusion detection datasets. For this, the

chapter proposes a number of requirements, and on this basis moves forward to proposing the Intrusion Detection Dataset Toolkit (ID2T). More specifically, ID2T offers an approach for injecting network files with cyber-attacks to generate labeled datasets. The results on the developed prototype suggest that the toolkit is able to handle large network files within a reasonable time period. Furthermore, ID2T generates datasets that do not include any parameters that might act as artifacts and thus degrade the quality of the generated dataset.

The state of the art was significantly improved with the introduction of the aforementioned chapters. First, the thesis introduces the concept of mobile honeypots along with novel detection mechanisms. This work also complements existing IDSs by providing signatures for the identification of attacks. Second, the proposed cyber incident monitor offers a platform for experimenting with various aspects of intrusion detection while offering an analysis of the lessons learned, from the deployment of such a system, for a long period of time. Finally, ID2T improves the alert data generation area by introducing an approach that moves beyond the logic of static datasets.

Collaborative Intrusion Detection

The third part of the thesis presents contributions in the core areas of collaborative intrusion detection. At a glance, Chapters 8 and 9 propose the communities concept and a fully distributed CIDS respectively. Chapter 10 contributes in the area of Probe Response Attacks (PRAs).

- The idea of *communities* of collaborative sensors is introduced in Chapter 8. The proposed CIDS concept makes use of communities of sensors that collaborate by exchanging features to create holistic and more accurate normality models for intrusion detection. The chapter additionally presents two stochastic algorithms for the creation of such communities. Moreover, the concept presented in this chapter is essentially the first to make use of alert data that are exchanged in the detection rather than the alert level. The results show the applicability of the concept and that the community-based approach is able to perform better than isolated intrusion detection and also provide accuracy levels that are similar to a centralized CIDS that exhibits global knowledge.

- On the basis of the aforementioned communities concept, Chapter 9 proposes *SkipMon* a fully distributed CIDS. *SkipMon* introduces a novel similarity-based correlation mechanism, on the basis of bloom filters, that can effectively correlate large amounts

of data towards creating communities of sensors. Furthermore, this is the first CIDS that supports domain awareness to dynamically constrain alert dissemination with respect to security policies. *SkipMon* exhibits a number of mechanisms for the propagation of information and it is also one of the first practical realizations of the SkipNet P2P overlay. The evaluation suggests that the CIDS can effectively create communities in an accuracy rate that is close to the one of a centralized system.

- Chapter 10, deals with the topic of PRAs (as introduced in Chapter 3) and proposes an open source framework, called *PREPARE*, for their deployment, experimentation and mitigation. On top of this, the chapter introduces major improvements in the design of a PRA and in the process of detecting such at attacks and reducing their impact. At a glance, the evaluation results first show the applicability of PRAs; the introduced proposals have been extensively tested in both a realistic simulation environment and a real world concept that involved attacking two different CIDSs. Furthermore, the comparison with the state of the art shows a significant improvement in terms of the time required for a successful execution of a PRA. Lastly, the proposed mitigation techniques appear to be highly effective, yet come with certain trade-offs.

Overall, the chapters, summarized above, improve the state of the art in three distinctive ways. First, the concept of communities can be utilized for CIDSs and the exchange of alert data in the detection level has been demonstrated. Second, the need for fulfilling the domain awareness requirement was addressed while the presented CIDS additionally exhibits novel correlation mechanisms. Lastly, the study and improvements proposed with regard to PRAs highlight the competence of such attacks and the need for further research with regard to them.

OUTLOOK

The thesis at hand offers distinctive improvements that advance the state of the art considerably. Nonetheless, further advancements in certain aspects of the presented topics can be envisioned. Here, the reader can find possible directions for improvements for selected topics in both the alert data creation and the collaborative intrusion detection part.

Alert Data Generation

HONEYPOT EVASION Honeypots provide a straightforward mechanism for detecting attacks, studying the adversaries techniques and

so forth. The work presented in Chapter 5 highlights, amongst other things, the importance, from both the perspective of the adversaries and the defenders, of detecting and evading honeypots. This topic has not received the necessary attention up to now and the rise of search engines, such as Shodan, accentuate the importance of stealthy, yet functional, honeypots. In fact, one could argue that the prototype utilized by Shodan (cf. Chapter 5.3) only makes use of simple techniques for the detection of honeypots.

Hence, further research can be conducted in both these directions. In the opinion of the author, the topic of circumventing detection is significantly challenging from the perspective of the honeypot. There is a trade off between publishing the source code of such a system openly, of offering enough interaction to the attacker and to applying techniques for remaining undetected. A starting point for such a system is a proper randomization of certain parameters. For instance, the analysis conducted in Chapter 5.3 illustrated that hard-coded and/or unusual parameters can lead to the disclosure of a honeypot.

From the perspective of an adversary, detecting honeypots can be considered somewhat straightforward. First, a study and analysis of the state of the art can serve as a basis for such a task. Usually, with regard to the topic or field that the malicious user is interested in, e.g., the IoT or a specific protocol, only a few honeypots exist. Furthermore, as transparency for such security projects is important, for the honeypot to be further utilized, the majority of honeypots are essentially open source projects. This provides the attacker the ability to extensively study the core parts of a honeypot and identify a possible misconfiguration and artifacts.

DATASET QUALITY In Chapter 7 the thesis examined the topic of datasets' quality in the context of evaluating intrusion detection algorithms and systems. Despite the efforts of this thesis and of other researchers, the area is still lacking a comprehensive survey of all existing datasets and of the respective toolkits for generating them. For such a task, an additional challenge that needs to be addressed is related to the availability of the proposals (e.g., many datasets are not publicly accessible).

On this basis, the next step can be the proposal of a formal model, towards providing an estimation of the quality of a given dataset. Similarly, such task is not trivial as one has to determine the level of influence that different parameters can have in a dataset. For instance, artifacts in the generation of TTL values might be more prominent than other parameters for the evaluation of a particular anomaly-based detection algorithm. Therefore, a qualitative comparison of the state of the art cannot be independent of the respective intrusion detection algorithms.

Collaborative Intrusion Detection

COLLABORATIVE INTRUSION DETECTION SYSTEMS (CIDSS) The discussion of CIDSs' related attacks in Chapter 3, highlighted the difficulties of detecting and handling insider attackers either on the host level or more importantly on the monitoring level. Existing work towards such a task (cf. Chapter 4) usually involves simple reputation systems and algorithms. Nevertheless, a holistic approach, that takes into account the state of the art in relevant fields, e.g., computational trust [137, 71, 7], is still to be investigated.

With regard to the core architectural area of CIDSs, it has become evident (cf. Chapter 4) that the decision of selecting the most appropriate CIDS class (i.e., centralized, hierarchical or distributed) is influenced by a number of parameters and it is not always straightforward to make such a determination. In this sense, the fully distributed approach presented in Chapter 9 can be enhanced into a more hybrid architecture. Such a decision can make it possible for the system to exchange alert data, or summaries of alert data, between communities from multiple network domains in an hierarchical manner. An approach towards such an architecture can be the election of community heads that are able to exchange data beyond the restrictions enforced by the domain awareness requirement.

PROBE RESPONSE ATTACKS (PRAS) The dissertation at hand made significant contributions in the area of PRAs. From the author's perspective future work in this direction may touch the following topics. First, further research can be conducted towards the applicability of PRAs. This can vary from performing practical attacks, e.g., via the utilization of the PREPARE framework (cf. Chapter 10), to create malware that are evasive by design. Note that real world testing of PRAs requires a lot of effort (e.g., one must be careful to avoid the blacklisting of research networks) and demands high resources (e.g., non-filtered traffic and also a significantly high volume bandwidth).

Moreover, the detection and mitigation of PRAs is also an area that contains research gaps and questions. The introduced detection methods in Chapter 10 are a first step towards an efficient detection of PRAs. Nevertheless, the following research questions remain.

- How can the PRAs be detected with a small false positive ratio?
- What kind of attacks can be mistakenly classified as PRAs?
- Does the dynamic nature of such datasets influence the selection of a threshold for the methods described in Chapter 10?

In addition, with regard to the mitigation of such attacks, the proposed adaptive reporting technique introduces a trade off between the quality of the data (as a result of the significant reduction of the

alert data due to sampling) and the mitigation itself. Hence, further research on evaluating the level of acceptable reduction of data in such a CIDS context is necessary.

Part V

APPENDIX

APPENDIX A - HOSTAGE FURTHER EVALUATION

More details on the evaluation of HosTaGe mobile honeypot are given in the following. First, Section A.1 shows a number of malware that have been detected with *HosTaGe*. Section A.2 deals with the topic of battery consumption.

HosTaGe Mobile Honeypot

MALWARE DETECTION IN HOSTAGE

Beyond the experiments shown in Chapter 5, the ability of *HosTaGe* to detect malware was further examined in a controlled environment. In more details, in order to examine the effectiveness of the system *HosTaGe* was deployed in an isolated testbed as shown in Figure 63. For this, several clients were connected to a wireless access point: a Linux-based system running a Dionaea honeypot, a Windows XP SP3 machine that was infected with a variety of malware (more details can be found in Table 14), and a mobile device with the following specifications:

- Device : Galaxy Nexus
- CPU: 1.2 GHz dual-core ARM Cortex-A9 (ARMv7 rev 10 [v7I])
- RAM: 693 MB
- OS: Android 4.2.2 (Jelly Bean)
- Mod-Version: CyanogenMod-10.1.2-maguro
- Linux-Kernel: 3.0.31

HosTaGe successfully detected all the attempts of the malware to propagate and the results also correspond to the reports from the Dionaea honeypot. Moreover, several port scans were also executed in the network using the Nmap network scanner[1], to test the resilience of *HosTaGe* and its effectiveness towards large number of port scans. *HosTaGe* successfully handled all these scans and remained functioning without any sign of misbehavior in the presence of large number of connection requests.

The malware used for the attack handling section were downloaded from the Open Malware Archive[2], operated by the Georgia Tech Information Security Center. Table 14 gives insights of the malware family as well as the MD5 hash of each particular malware that was utilized and successfully detected.

1 http://www.nmap.org
2 http://oc.gtisc.gatech.edu

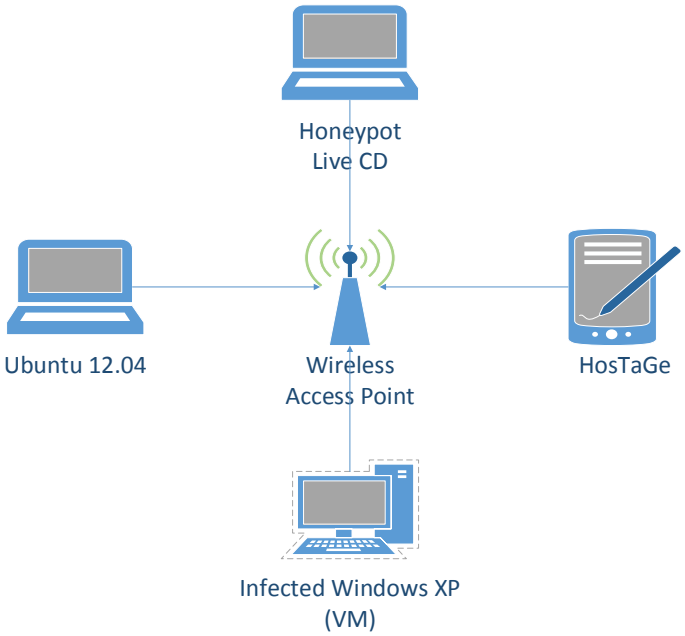


Figure 63: Attack Testbed Architecture.

Malware Family	MD5 Hash
Win32/Themida	682d6c26a81c5f62e9bb02349c804995
Worm/Generic.AHB	e56d66fa40e3c2097b8824953a5250cb
Downloader.Generic10.BNH	9523e99c4d4267f813c97e795b33480e
PSW.Generic8.GKS	7fe658710e4904e0ee2148ddee02b8e9
PSW.Generic8.HBB	d29a563bdca54d8ca381ea75ff619b2d
PSW.OnlineGames3.AOPH	41b3419b105afe85a38a3ca2765c53fd
Worm/Delf.HA	2e83efa9412ada82f527005d44281792
Worm/Delf.LC	3947c2e31e87ae4a6d5c81b3cee14b15
Win32/Prepender.J	21dc5b946d7a09999f205410ad8f080e

Table 14: Malware Deployed In The Testbed.

BATTERY CONSUMPTION

While the previous experiments highlight the ability of *HosTaGe* to detect attacks, one of the basic requirements of any mobile application is to provide a substantive power utilization. The performance of *HosTaGe* was profiled using an Android application called PowerTutor[189]. PowerTutor provides measurements of power utilization of an application on Android OS. The utilization of *HosTaGe* was firstly com-

pared with other frequently used applications such as WhatsApp[3], Facebook [4] and the AVG Free AntiVirus[5].

To measure the honeypot's power utilization, *HosTaGe* was deployed (running in the background) in a network and an automated script for testing was executed. The same device as described in the attack detection analysis (above) was utilized. The script automates random number of protocol connections, between none to five connections every 30 seconds, to the *HosTaGe* device (emulating attack) for a total duration of 60 minutes.

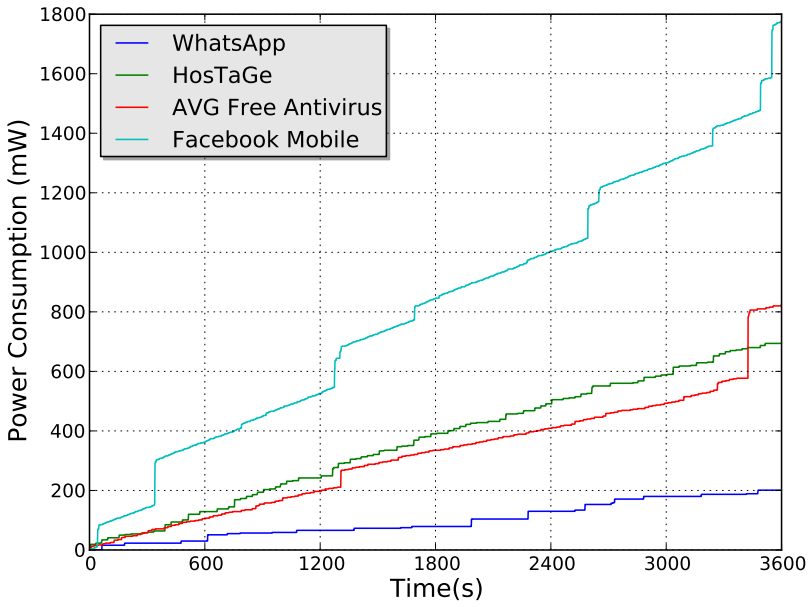


Figure 64: Power consumption of HosTaGe in comparison to other applications.

The power consumption of *HosTaGe* (with script automation) was compared to the other Android applications which were executed in the background. The resulting comparison graph is shown in Figure 64. As depicted in the figure, *HosTaGe* performs reasonably well considering the amount of attacks handled by it throughout the testing duration.

Finally, Figure 65 shows a comparison of different deployments of the honeypot. In more details, the following three different settings are utilized:

- *HosTaGe* working in the background with the multistage detection functionality deactivated

3 http://www.whatsapp.com
4 https://www.facebook.com/mobile/
5 http://www.avg.com/eu-en/antivirus-for-android

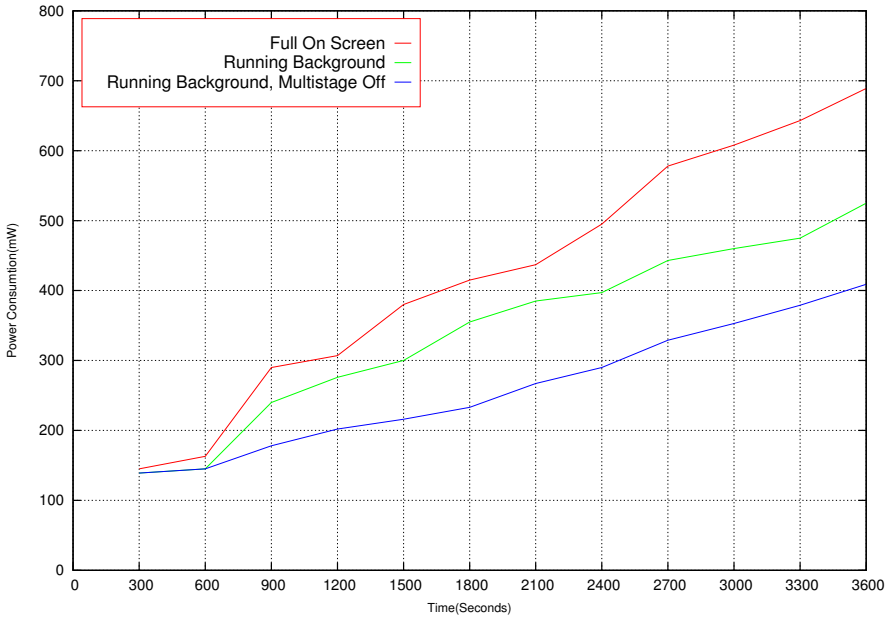


Figure 65: Power consumption of HosTaGe for various settings.

- *HosTaGe* working in the background with the multistage detection functionality activated
- *HosTaGe* working on the screen and with the multistage detection functionality activated

As depicted in Figure 65, there is an increase in the battery consumption when the honeypot is also utilizing the screen. Furthermore, the multistage detection functionality introduces a rather small increase of battery consumption.

B

APPENDIX B - SKIPNET BACKGROUND

SKIPNET

SkipNet [67] is an extension of SkipLists [128] for P2P networking. In this context two approaches have been proposed, i.e., Skip Graphs [8] and SkipNet [67]. We utilize the latter one as it provides features and details that are useful in a practical manner. Some of the notable properties that SkipNet offers are the routing tables that take into account the link quality between nodes and the network maintenance mechanisms that take place after major network disruptions.

In SkipNet all participating nodes are placed in a ring and identified with their reversed DNS name. In a practical realization this can be utilized to group nodes (in our case monitoring *sensors*) of the same organization or sub-network, as their hostnames end with the same domain names and their identifiers will start with the same prefix. Each sensor will not only have a link to their next neighbor but also to nodes 2^n hops away, similarly to the so-called *fingers* in Chord [157]. In this case, however, the nodes form sub-rings with the higher level links, as shown in Figure 66. For each routing level, nodes choose randomly the sub-ring that they will join. Their ring will lead to a second identifier, called numeric ID. Similar to SkipLists, each SkipNet node can hold data, which can be identified with a Uniform Resource Identifier (URI) providing the location and the name of the resource.

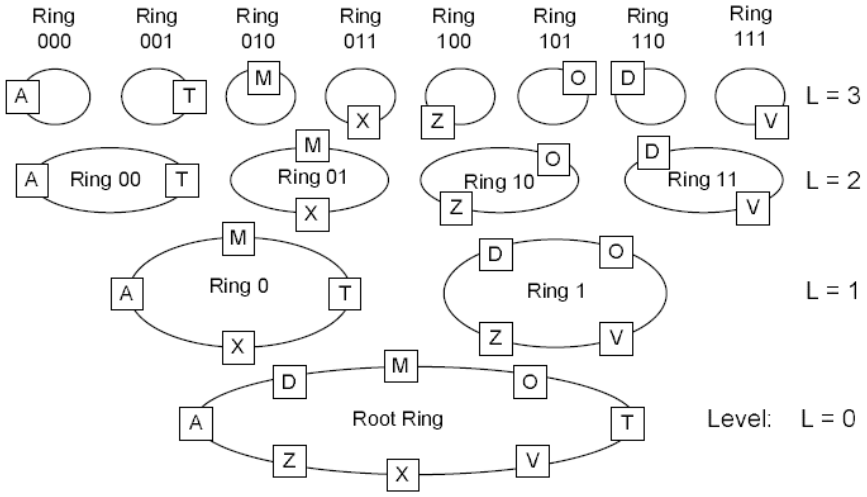


Figure 66: SkipNet routing infrastructure example [67]

One of the benefits of SkipNet is the possibility to achieve *data locality* and *domain awareness*. When storing data in SkipNet, similarly to a DHT, data locality can be achieved. When specifying one node in the URI for a resource, the resource is stored at this node. Afterwards, the exact location of the resource is known. This is usually impossible when using load balancing. However, with the Constrained Load Balancing (CLB) in SkipNet, a domain can be specified, and the resource will be stored at a member of this domain. Thus, the location of the resource can be limited to groups of nodes; providing what is defined in Chapter 9 as *domain awareness*.

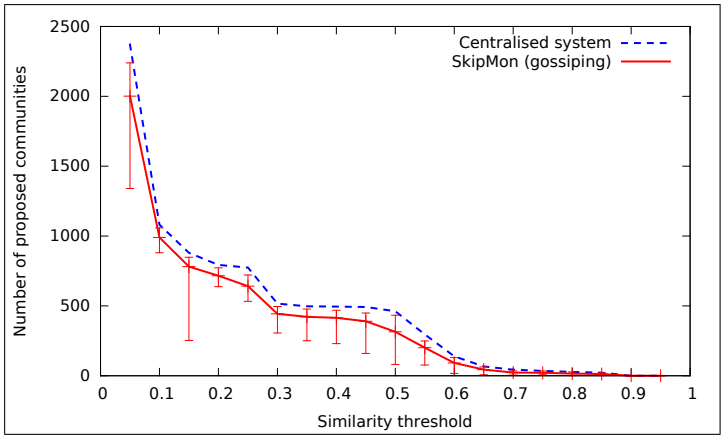
However, for *SkipMon* (see Chapter 9), data will be forwarded between the system instead of storing it at a given node or set of nodes. Thus, in this work, only the *domain awareness* of SkipNet is important. Due to the routing algorithms and ordering of nodes in SkipNet, data that shall be exchanged in one domain can and will only be routed through nodes of this domain. This leads to implicit data locality, as data transferred between two nodes in the same domain will never leave the boundaries of the domain in transit.

C

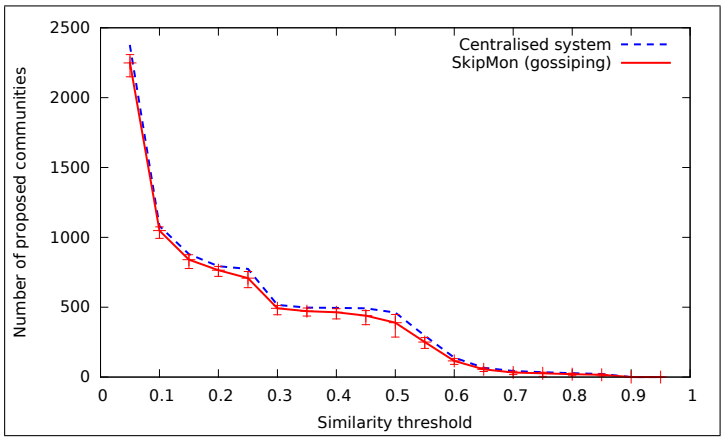
APPENDIX C - SKIPMON EVALUATION

SKIPMON FURTHER EVALUATION

In the following further results of the evaluation of *SkipMon* are given. In more details, the following experiments are an extension of Chapter 9.4.3.



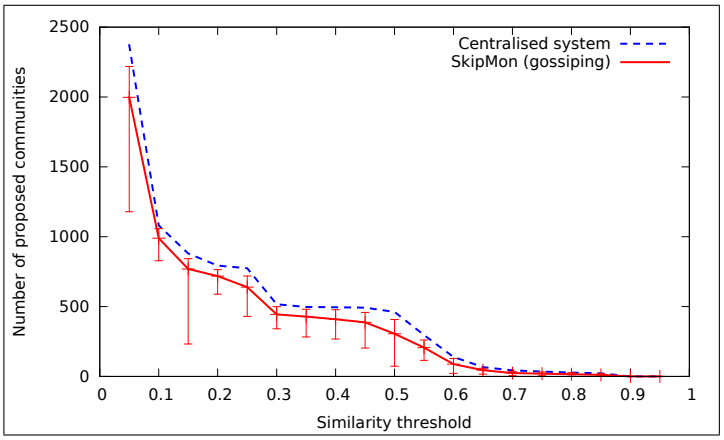
(a) Proposed communities by *SkipMon* (with gossiping, $k = 5$) compared to a centralized system.



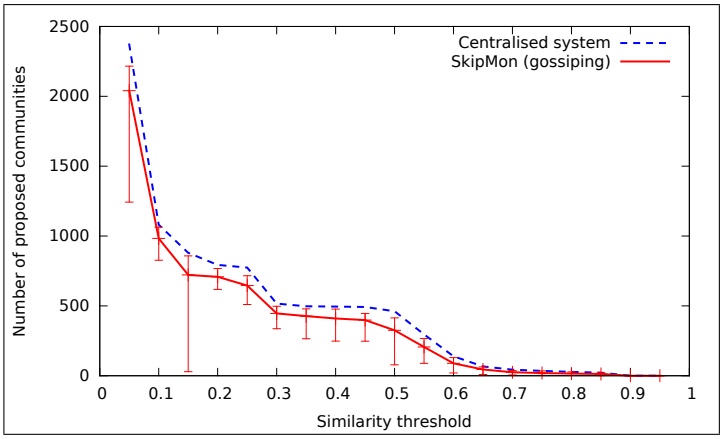
(b) Proposed communities by *SkipMon* (with gossiping, $k = 6$) compared to a centralized system.

Figure 67: Proposed communities by *SkipMon* (with gossiping and k between 5 and 6) compared to a centralized system.

In particular, Figures 67 and 68 correspond to various cases, similar to Figure 42 (cf. Chapter 9), but by modifying the value k of Equation 2 (see Section 9.2.3). In essence, this translates into a lower probability for the gossiping. Nevertheless, as it is suggested by the experiments



(a) Proposed communities by *SkipMon* (with gossiping, $k = 7$) compared to a centralized system.



(b) Proposed communities by *SkipMon* (with gossiping, $k = 8$) compared to a centralized system.

Figure 68: Proposed communities by *SkipMon* (with gossiping and k between 7 and 8) compared to a centralized system.

even though the probability for the gossiping algorithm is decreasing (with a smaller value k) the accuracy of the system (in comparison to the centralized approach) is only slightly decreased.

BIBLIOGRAPHY

[1] Caida: The cooperative association for internet data analysis. http://www.caida.org, 2015.

[2] Seyed Hossein Ahmadinejad and Saeed Jalili. Alert correlation using correlation probability estimation and time windows. In *Computer Technology and Development, 2009. ICCTD'09. International Conference on*, volume 2, pages 170–175. IEEE, 2009.

[3] Eugene Albin and Neil C. Rowe. A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems. In *26th International Conference on Advanced Information Networking and Applications Workshops*, pages 122–127. IEEE, mar 2012. ISBN 978-1-4673-0867-0.

[4] Stephanos Androutsellis-theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004.

[5] S Antonatos, E P Markatos, and K G Anagnostakis. Honey @ home : A New Approach to Large-Scale Threat Monitoring. In *ACM workshop on Recurring malcode*, pages 38–45. ACM, 2007. ISBN 9781595938862.

[6] Spiros Antonatos, Michael Locasto, and Stelios Sidiroglou. Defending Against Next Generation through Network / Endpoint Collaboration and Interaction. In *3rd European Conference on Computer Network Defense*, pages 131–141. Springer US, 2009.

[7] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, 2007. ISSN 15708268. doi: 10.1016/j.websem.2007.03.002.

[8] James Aspnes and Gauri Shah. Skip graphs. *ACM Transactions on Algorithms*, 3(4):37, nov 2007. ISSN 15496325. doi: 10.1145/1290672.1290674.

[9] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Department of Computer Engineering, Chalmers University, 2000.

[10] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The nepenthes platform: An efficient approach to collect malware. *Lecture notes in Computer Science*, 4219:165–184, 2006.

[11] Jai Sundar Balasubramaniyan, Jose Omar Garcia-fernandez, David Isacoff, Eugene Spafford, and Diego Zamboni Ý. An architecture for intrusion detection using autonomous agents. In *IEEE computer security applications conference*, pages 13–24, 1998.

[12] Bazara I A Barry and H Anthony Chan. Intrusion Detection Systems. In *Handbook of Information and Communication Security*, pages 193–205. Springer Berlin, 2010.

[13] John Bethencourt, J Franklin, and M Vernon. Mapping internet sensors with probe response attacks. In *USENIX Security Symposium*, pages 193–208, 2005.

[14] Philippe Biondi. Network packet manipulation with scapy, 2007.

[15] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection*, 7(2):114–123, 2014.

[16] Ali Borji. Combining heterogeneous classifiers for network intrusion detection. In *Advances in Computer Science–ASIAN 2007. Computer and Network Security*, pages 254–260. Springer, 2007.

[17] Daniela Brauckhoff, Arno Wagner, and May Martin. FLAME: a Flow-Level Anomaly Modeling Engine. In *The conference on Cyber security (CSET)*, 2008.

[18] Michael Brinkmeier, Mathias Fischer, Sascha Grau, and Guenter Schaefer. Towards the Design of Unexploitable Construction Mechanisms for Multiple-Tree Based P2P Streaming Systems. In *Kommunikation in Verteilten Systemen (KiVS)*, pages 193–204. Springer Berlin Heidelberg, 2009.

[19] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485–509, jan 2004. ISSN 1542-7951.

[20] Ismail Butun, Salvatore D. Morgera, and Ravi Sankar. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 16(1):266–282, 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.050113.00191.

[21] Rainer Bye, Seyit Ahmet Campete, and Sahin Albayrak. Collaborative Intrusion Detection Framework : Characteristics , Adversarial Opportunities and Countermeasures. In *Workshop on Collaborative Methods for Security and Privacy (CollSec)*, pages 1–12, 2010.

[22] Yu Chen Cai, Min, Kai Hwang, Yu-Kwong Kwok, Shanshan Song. Collaborative Internet Worm Containment. *IEEE Security and Privacy Magazine*, 3(3):25–33, may 2005. ISSN 1540-7993.

[23] Antony I.T. Castro, Miguel, Druschel, Peter, Kermarrec, A.-M., Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, oct 2002. ISSN 0733-8716.

[24] Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[25] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A Survey. *ACM Computing Surveys*, 41(3): 1–58, jul 2009. ISSN 03600300.

[26] Tsung-huan Cheng, Y Lin, Yuan-cheng Lai, and Po-ching Lin. Evasion Techniques: Sneaking through Your Intrusion Detection/Prevention Systems. *IEEE Communications Surveys & Tutorials*, (99):1–10, 2011.

[27] S Cheung. Securing collaborative intrusion detection systems. *IEEE Security and Privacy*, 9(6):36–42, 2011.

[28] Steven Cheung, Rick Crawford, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, Je Rowe, Stuart Staniford-chen, Raymond Yip, and Dan Zerkle. The Design of GrIDS : A Graph-Based Intrusion Detection System. Technical report, University of California at Davis, 1999.

[29] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Nikolay Milanov, Christian Koch, David Hausheer, and Max Mühlhäuser. ID2T: a DIY dataset creation toolkit for Intrusion Detection Systems. In *Conference on Communications and Network Security (CNS)*, pages 739–740. IEEE, 2015. ISBN 9781467378765.

[30] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Max Mühlhäuser, and Mathias Fischer. Community-based Collaborative Intrusion Detection. In *International Workshop on Applications and Techniques in Cyber Security*. Springer, 2015.

[31] Gideon Creech and Jiankun Hu. Generation of a new IDS Test Dataset : Time to Retire the KDD Collection. In *Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013. ISBN 9781467359399.

[32] Mark Crosbie, B Dole, T Ellis, Ivan Krsul, and EH Spafford. Idiot-users guide. Technical report, 1996.

[33] Frédéric Cuppens. Managing alerts in a multi-intrusion detection environment. In *Annual Computer Security Applications*, pages 22–31. IEEE, 2001. ISBN 0-7695-1405-7.

[34] Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2002.

[35] Zoltán Czirkos and Gábor Hosszú. Enhancing Collaborative Intrusion Detection Methods Using a Kademlia Overlay Network. In *18th EUNICE/ IFIP WG 6.2, 6.6 International Conference*, volume 7479, pages 52–63. Springer, 2012.

[36] Oliver Dain and Robert K Cunningham. Fusing a Heterogeneous Alert Stream into Scenarios. In *ACM workshop on data mining for security applications*, pages 1–13, 2001.

[37] Herve Debar and Andreas Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Recent Advances in Intrusion Detection*, pages 85–103. Springer, 2001.

[38] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31 (8):805–822, apr 1999. ISSN 13891286.

[39] Herve Debar, David A. Curry, and Benjamin S. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF), 2007.

[40] John R Douceur. The sybil attack. In *Peer-to-Peer Systems*, pages 251–260. Springer Berlin Heidelberg, 2002.

[41] Claudiu Duma, Martin Karresand, Nahid Shahmehri, and Germano Caronni. A Trust-Aware, P2P-Based Overlay for Intrusion Detection. In *International Conference on Database and Expert Systems Applications (DEXA'06)*, pages 692–697. IEEE, 2006. ISBN 0-7695-2641-1.

[42] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Proceedings of the 22nd USENIX Security Symposium*, pages 605–619, 2013. ISBN 9781931971034.

[43] Kevin P Dyer, Scott E Coull, and Thomas Shrimpton. Marionette: A programmable network traffic obfuscation system. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 367–382, 2015.

[44] Steven T. Eckmann, Giovanni Vigna, and Richard A. Kemmerer. STATL: An attack language for state-based intrusion detection. *Journal of Computer Security*, 10(1-2):71–103, 2002.

[45] Sven Ehlert, Dimitris Geneiatakis, and Thomas Magedanz. Survey of network security systems to counter sip-based denial-of-service attacks. *Computers & Security*, 29(2):225–243, 2010.

[46] David Ehringer. The dalvik virtual machine architecture. *Techn. report (March 2010)*, 4, 2010.

[47] Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman. Alert correlation in collaborative intelligent intrusion detection systems: A survey. *Applied Soft Computing*, 11(7):4349–4365, oct 2011. ISSN 15684946.

[48] Vasilomanolakis Emmanouil and Krügl Matthias. SkipMon collaborative intrusion detection system. http://scm.tk.informatik.tu-darmstadt.de/projects/scm-ssi-skipmon.

[49] Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications*, 27(16):1569–1584, oct 2004. ISSN 01403664.

[50] Marcelo Fantinato and Mario Jino. Applying extended finite state machines in software testing of interactive systems. In *Interactive Systems: Design, Specification, and Verification (DSV-IS)*, volume 2844, pages 34–45. Springer Berlin Heidelberg, 2003. ISBN 03029743 (ISSN).

[51] Prahlad Fogla, Monirul I. Sharif, Roberto Perdisci, Oleg M. Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *USENIX Security Symposium*, pages 241–256, 2006.

[52] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 1–12. ACM, 2010.

[53] Carol Fung. Collaborative intrusion detection networks and insider attacks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):63–74, 2011.

[54] Carol Fung, Olga Baysal, Jie Zhang, Issam Aib, and Raouf Boutaba. Trust management for host-based collaborative intrusion detection. *Managing Large-Scale Service Deployment*, 5273: 109–122, 2008.

[55] Carol J. Fung, Jie Zhang, Issam Aib, and Raouf Boutaba. Robust and scalable trust management for collaborative intrusion detection. In *International Symposium on Integrated Network Management*, pages 33–40. IEEE, jun 2009. ISBN 978-1-4244-3486-2.

[56] Antonio Galante, Ary Kokos, Stefano Zanero, and Politecnico Milano. BlueBat : Towards Practical Bluetooth Honeypots. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2009. ISBN 9781424434350.

[57] Ayalvadi J. Ganesh, A-M. Kermarrec, and Laurent Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, feb 2003. ISSN 0018-9340.

[58] Joaquin Garcia, Fabien Autrel, Joan Borrell, Sergio Castillo, Frederic Cuppens, and Guillermo Navarro. Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation. In *Information and Communications Security*, pages 223–235. Springer, 2004.

[59] Pedro Garcia-Teodoro, J. Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, feb 2009. ISSN 01674048.

[60] Vangelis Gazis, Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Panayotis Kikiras, and Alex Wiesmaier. Security Perspectives for Collaborative Data Acquisition in the Internet of Things. In *International Conference on Safety and Security in Internet of Things*. Springer, 2015.

[61] Vangelis Gazis, Manuel Görtz, Marco Huber, Alessandro Leonardi, Kostas Mathioudakis, Alexander Wiesmaier, Florian Zeiger, and Emmanouil Vasilomanolakis. A Survey of Technologies for the Internet of Things. In *International Wireless Communications & Mobile Computing Conference (IWCMC), Machine - to - Machine Communications (M2M) & Internet of Things (IoT) Workshop*. IEEE, 2015.

[62] Manuel Gil Pérez, Félix Gómez Mármol, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. RepCIDN: A Reputation-based Collaborative Intrusion Detection Network to Lessen the Impact of Malicious Alarms. *Journal of Network and Systems Management*, 21(1):128–167, mar 2013. ISSN 1064-7570.

[63] Jan; Gobel. Amun : Automatic Capturing of Malicious Software. Technical report, University of Mannheim, 2010.

[64] Li Gong. JXTA: A network programming environment. *Internet Computing, IEEE*, 5(3):88–95, 2001.

[65] John R Goodall, Wayne G Lutters, and Anita Komlodi. I Know My Network: Collaboration and Expertise in Intrusion Detection. In *ACM conference on Computer supported cooperative work*, pages 342–345. ACM, 2004. ISBN 1581138105.

[66] M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. Pervasive Computing. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[67] Nicholas JA Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, volume 4, pages 1–14, Seattle, WA, 2003. USENIX Association.

[68] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and Survey of the Visible Internet. In *Proceedings of the ACM Internet Measurement Conference*, pages 169–182, 2008. ISBN 9781605583341. doi: 10.1145/1452520.1452542.

[69] Mark D Hill. What is scalability? *ACM SIGARCH Computer Architecture News*, 18(4):18–21, 1990.

[70] Ramaprabhu Janakiraman, Marcel Waldvogel, and Qi Zhang. Indra: a peer-to-peer approach to network intrusion detection and prevention. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, pages 226–231. IEEE, 2003. ISBN 0-7695-1963-6.

[71] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007. ISSN 01679236. doi: 10.1016/j.dss.2005.05.019.

[72] Peyman Kabiri and Ali A Ghorbani. Research on Intrusion Detection and Response : A Survey. *International Journal of Network Security*, 1(2):84–102, 2005.

[73] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, page 640, 2003.

[74] Pradeep Kannadiga and Mohammad Zulkernine. DIDMA : A Distributed Intrusion Detection System Using Mobile Agents. In *International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 238 – 245. IEEE, 2005. ISBN 0769522947.

[75] Shankar Karuppayah, Emmanouil Vasilomanolakis, Steffen Haas, Max Mühlhäuser, and Mathias Fischer. BoobyTrap: On Autonomously Detecting and Characterizing Crawlers in P2P Botnets. In *International Conference on Communications (ICC)*. IEEE, 2016.

[76] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*, 41 (5):2, 2007. ISSN 01635980. doi: 10.1145/1317379.1317381.

[77] Anne Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3): 248–258, 2003. ISSN 10459219. doi: 10.1109/TPDS.2003.1189583.

[78] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX security symposium*, volume 286. San Diego, CA, 2004.

[79] Robert Koch, Mario Golling, and Gabi Dreo Rodosek. Towards Comparability of Intrusion Detection Systems: New Data Sets. In *TERENA Networking Conference*, page 7, 2014.

[80] Christian Kreibich and Jon Crowcroft. Honeycomb: Creating Intrusion Detection Signatures Using Honeypots. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 51 – 56, 2004.

[81] Christopher Kruegel, Engin Kirda, and Darren Mutz. Automating mimicry attacks using static binary analysis. In *USENIX Security Symposium*, pages 161–176, 2005.

[82] Christopher Krügel, Thomas Toth, and Clemens Kerer. Decentralized event correlation for intrusion detection. In *Information Security and Cryptology (ICISC)*, volume 2288, pages 114–131. Springer Berlin Heidelberg, 2002.

[83] Christopher Krugel, Thomas Toth, and Engin Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. In *ACM symposium on Applied computing (SAC)*, pages 201–208. ACM, 2002. ISBN 1581134452.

[84] Matthias Krügl. *Memberhip Management for unstructured distributed Collaborative IDS*. Master thesis, Technische Universität Darmstadt, 2015.

[85] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, oct 1973. ISSN 00010782.

[86] R Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, (June):49–51, 2011.

[87] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. Intrusion Detection: A Survey. In *Managing Cyber Threats*, volume 5, pages 19–78. Springer US, 2005.

[88] Zhichun Li, Yan Chen, and Aaron Beach. Towards scalable and robust distributed intrusion alert fusion with good load balancing. In *SIGCOMM workshop on Large-scale attack defense (LSAD)*, pages 115–122, New York, New York, USA, 2006. ACM Press. ISBN 1595935711.

[89] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1): 16–24, 2013. ISSN 10848045. doi: 10.1016/j.jnca.2012.09.004.

[90] Steffen Liebergeld, Matthias Lange, and Collin Mulliner. Nomadic Honeypots : A Novel Concept for Smartphone Honeypots. In *Workshop on Mobile Security Technologies (MoST), in conjunction with the 34th IEEE Symp. on Security and Privacy.*, 2013.

[91] P Lincoln, P Porras, and V Shmatikov. Privacy-preserving sharing and correction of security alerts. In *13th USENIX Security Symposium*, pages 239–254, 2004.

[92] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[93] Michael E. Locasto, Janak J. Parekh, Salvatore Stolfo, and Vishal Misra. Collaborative distributed intrusion detection. Technical report, Columbia University, 2004.

[94] Michael E. Locasto, Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo. Towards Collaborative Security and P2P Intrusion Detection. In *IEEE Workshop on Information Assurance and Security*, pages 333 – 339. IEEE, 2005. ISBN 0780392906.

[95] MEA Lopez and OCMB Duarte. Providing elasticity to intrusion detection systems in virtualized software defined networks. In *IEEE ICC*, 2015.

[96] EK Lua, J Crowcroft, and M Pias. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 7(2):72–93, 2005.

[97] Jie Ma, Zhi-tang Li, and Wei-ming Li. Real-time alert stream clustering and correlation for discovering attack strategies. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 4, pages 379–384. IEEE, 2008.

[98] Dirk Maan, José Jair Santanna, Anna Sperotto, and Pietertjerk De Boer. Towards validation of the Internet Census 2012. In *20th EUNICE/IFIP EG 6.2, 6.6 International Workshop*, pages 85–96. Springer, 2014.

[99] Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. *Journal Of Artificial Intelligence Research*, 11: 169–198, 1999.

[100] Matthew V Mahoney and Philip K Chan. Learning Models of Network Traffic for Detecting Novel Attacks, 2002.

[101] Matthew V Mahoney and Philip K Chan. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. *International Symposium on Recent Advances in Intrusion Detection*, 2820:220–237, 2003. ISSN 0302-9743. doi: 10.1007/b13476.

[102] Matthew V. MV Mahoney and PK Philip K. Chan. PHAD: Packet Header Anomaly Detection for identifying hostile network traffic. Technical Report 1998, Florida Institute of Technology, 2001.

[103] M.V. Mahoney and P.K. Chan. Learning rules for anomaly detection of hostile network traffic. In *IEEE International Conference on Data Mining*, pages 601–604. IEEE Comput. Soc, 2003. ISBN 0-7695-1978-4.

[104] Mirco Marchetti, Michele Messori, and Michele Colajanni. Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale. *Lecture Notes in Computer Science*, 5735:475–490, 2009.

[105] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4): 472–484, mar 2006. ISSN 13891286.

[106] J McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM transactions on Information and system Security*, 3(4):262–294, 2000.

[107] Guozhu Meng, Yang Liu, Jie Zhang, Alexander Pokluda, and Raouf Boutaba. Collaborative security: A survey and taxonomy. *ACM Comput. Surv.*, 48(1):1:1–1:42, July 2015. ISSN 0360-0300. doi: 10.1145/2785733.

[108] Nikolay Milanov. *ID2T: an Intrusion Detection Dataset Toolkit*. Master thesis, Technische Universität Darmstadt, 2015.

[109] Yin Pa Minn, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, and Christian Rossow. IoTPOT : Analysing the Rise of IoT Compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2015.

[110] SA Mirheidari, Sajjad Arshad, and Rasool Jalili. Alert Correlation Algorithms: A Survey and Taxonomy. *Cyberspace Safety and Security*, pages 183–197, 2013.

[111] J Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communications Review*, 34(2):39–53, 2004.

[112] Robert Mitchell and Ing-ray Chen. A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Computing Surveys (CSUR)*, 46(4):55:1–29, 2014. ISSN 03600300. doi: 10.1145/2542049.

[113] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[114] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*, 36(1):42–57, 2013. ISSN 10848045. doi: 10.1016/j.jnca.2012.05.003.

[115] David Moore, Colleen Shannon, and Jeffery Brown. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. In *Second ACM SIGCOMM Workshop on Internet Measurment (IMW)*, pages 273–284, 2002. ISBN 158113603X. doi: 10.1145/637201.637244.

[116] Collin Mulliner, Steffen Liebergeld, and Matthias Lange. Poster : HoneyDroid - Creating a Smartphone Honeypot. In *IEEE Symposium on Security and Privacy (S&P)*, 2011.

[117] Mohamed Nassar, Radu State, and Olivier Festor. Voip honeypot architecture. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 109–118. IEEE, 2007.

[118] Milanov Nikolay, Vasilomanolakis Emmanouil, and Garcia Cordero Carlos. The id2t: Intrusion detection dataset toolkit. https://www.tk.informatik.tu-darmstadt.de/de/research/secure-smart-infrastructures/id2t/, 2015.

[119] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

[120] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, dec 1999. ISSN 13891286.

[121] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer networks*, 53 (6):864–881, 2009.

[122] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, nov 2012. ISSN 2192-6352.

[123] Eric Peter and Todd Schiller. A Practical Guide to Honeypots. Technical report, Washington Univerity, 2011.

[124] Phillip A. Porras and Peter G. Neumann. EMERALD: Event monitoring enabling response to anomalous live disturbances. In *National information systems security conference (NISSC)*, pages 353–365, 1997.

[125] Phillip A Porras, Martin W Fong, and Alfonso Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *Conference on Recent Advances in Intrusion Detection (RAID)*, pages 95–114. Springer, 2002.

[126] Georgios Portokalidis, Asia Slowinska, and Herbert Bos. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. *ACM SIGOPS Operating Systems Review*, 40(4):15–27, 2006.

[127] Niels Provos. Honeyd : A Virtual Honeypot Daemon. In *DFN-CERT workshop*, 2003.

[128] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990. ISSN 00010782. doi: 10.1145/78973.78977.

[129] Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Fast and Evasive Attacks: Highlighting the challenges ahead. In *Recent Advances in Intrusion Detection*, volume 4219, pages 206–225. Springer Berlin Heidelberg, 2006.

[130] Geetha Ramachandran and Delbert Hart. A P2P Intrusion Detection System based on Mobile Agents. In *Southeast regional conference ACM-SE*, pages 185–190. ACM, 2004. ISBN 1581138709.

[131] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[132] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *USENIX Annual Techincal Conference*, number June, pages 127–140, 2004.

[133] Sean Rhea, Brighten Godfrey, and Brad Karp. OpenDHT: a public DHT service and its uses. *ACM SIGCOMM Computer Communication Review*, 35(4):73–84, 2005.

[134] Lukas Rist, Daniel Haslinger, John Smith, Johny Vestergaard, and Andrea Pasquale. Conpot honeypot, 2013.

[135] Martin Roesch. Snort-lightweight intrusion detection for networks. In *USENIX conference on System administration*, pages 229–238, 1999.

[136] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Middleware 2001*, pages 329–350, 2001.

[137] Jordi Sabater and Carles Sierra. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60, 2005. ISSN 0269-2821. doi: 10.1007/s10462-004-0041-5.

[138] Saeed Salah, Gabriel Maciá-Fernández, and Jesús E. Díaz-Verdejo. A model-based survey of alert correlation techniques. *Computer Networks*, 57(5):1289–1317, 2013. ISSN 13891286. doi: 10.1016/j.comnet.2012.10.022.

[139] Benjamin Sangster, Thomas Cook, Robert Fanelli, Erik Dean, William J Adams, Chris Morrell, and Gregory Conti. Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In *USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET)*, 2009.

[140] Christian Seifert, Ian Welch, and Peter Komisarczuk. HoneyC - The Low-Interaction Client Honeypot. In *NZCSRCS*, 2007.

[141] Poly Sen, Nabendu Chaki, and Rituparna Chaki. HIDS: Honesty-Rate Based Collaborative Intrusion Detection System for Mobile Ad-Hoc Networks. In *7th Computer Information Systems and Industrial Management Applications*, pages 121–126. IEEE, jun 2008. ISBN 978-0-7695-3184-7.

[142] S. Shin and Guofei Gu. Conficker and beyond: a large-scale empirical study. In *26th Annual Computer Security Applications Conference*, pages 151–160, 2010. ISBN 9781450301336. doi: 10.1145/1920261.1920285.

[143] Yoichi Shinoda, K Ikai, and M Itoh. Vulnerabilities of passive internet threat monitors. In *USENIX Security Symposium*, pages 209–224, 2005.

[144] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali a. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[145] Vitaly Shmatikov and Ming-Hsiu Wang. Security against probe-response attacks in collaborative intrusion detection. In *Workshop on Large scale attack defense - LSAD*, pages 129–136, New York, New York, USA, 2007. ACM. ISBN 9781595937858.

[146] Steven Snapp, James Brentano, Gihan Dias, Terrance Goan, Todd Heberlein, Che-Lin Ho, Karl Levitt, Biswanath Mukherjee, Stephen Smaha, Tim Grance, Daniel Teal, and Doug Mansur. DIDS (Distributed intrusion detection system) - Motivation, Architecture, and an early Prototype. In *Fourteenth National Computer Security Conference*, pages 167–176, 1991.

[147] Tomas Sochor and Matej Zuzcak. Study of internet threats and attack methods using honeypots and honeynets. In *Computer Networks*, pages 118–127. Springer, 2014.

[148] Aditya K. Sood and Richard J. Enbody. Targeted Cyber Attacks-A Superset of Advanced Persistent Threats. *IEEE Security & Privacy*, 11(1):54–61, 2013.

[149] Eugene H Spafford and Diego Zamboni. Intrusion Detection using Autonomous Agents. *Computer Networks*, 34(4):547–570, 2000.

[150] Lance Spitzner. *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading, 2003.

[151] Lance Spitzner. Honeypots : Catching the Insider Threat. In *Computer Security Applications Conference*, pages 170–179. IEEE, 2003.

[152] Shreyas Srinivasa. *A Mobile Honeypot for Industrial Control Systems*. Master thesis, Technische Universität Darmstadt, 2015.

[153] A. Srivastava, B. B. Gupta, A. Tyagi, Anupama Sharma, and Anupama Mishra. A recent survey on DDoS attacks and defense mechanisms. In *International Conference on Parallel, Distributed Computing Technologies and Applications*, volume 203 CCIS, pages 570–580. Springer Berlin Heidelberg, 2011. ISBN 9783642240362. doi: 10.1007/978-3-642-24037-9{_}57.

[154] Michael Stahn. *Probe response attacks on cyber incident monitors*. Master thesis, Technische Universität Darmstadt, 2015.

[155] Staniford-Chen, Steven Cheung Stuart, Richard Crawford, Mark Dilger, Jeremy Frank, James Hoagland, Karl Levitt, Christopher Wee, Raymond Yip, and Dan Zerkle. GrIDS - A Graph Based Intrusion Detection System for Large Networks. In *National information systems security conference*, pages 361–370, 1996.

[156] Peter Stavroulakis and Mark Stamp. *Handbook of information and communication security*. Springer Science & Business Media, 2010.

[157] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan Ý. Chord : A Scalable Peer-to-peer Lookup Service for Internet. In *Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM, 2001. ISBN 1581134118.

[158] Jonathan Stone, Michael Greenwald, Craig Partridge, Senior Member, and James Hughes. Performance of Checksums and CRCs over Real Data. *October*, 6(5):529–543, 1998.

[159] Kymie MC Tan, Kevin S. Killourhy, and Roy A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection*, volume 2516, pages 54–73. Springer Berlin Heidelberg, 2002.

[160] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and Practice of Bloom Filters for Distributed Systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2012. ISSN 1553-877X. doi: 10.1109/SURV.2011.031611.00024.

[161] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Symposium on Computational Intelligence for Security and Defense Applications, CISDA*, number Cisda, pages 1–6. IEEE, 2009. ISBN 9781424437641. doi: 10.1109/CISDA.2009.5356528.

[162] Urjita Thakar, Sudarshan Varma, and AK Ramani. Honeyanalyzer–analysis and extraction of intrusion detection patterns & signatures using honeypot. In *Proceedings of the Second International Conference on Innovations in Information Technology*, 2005.

[163] Stefan Tilkov and Steve Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

[164] Virus Total. Virustotal-free online virus, malware and url scanner, 2012.

[165] Johannes Ullrich. Dshield internet storm center. https://www.dshield.org/, 2000.

[166] Alfonso Valdes and Keith Skinner. Probabilistic Alert Correlation. In *Recent Advances in Intrusion Detection*, pages 54–68. Springer, 2001.

[167] Emmanouil Vasilomanolakis, Mathias Fischer, Max Mühlhäuser, Peter Ebinger, Panayotis Kikiras, and Sebastian Schmerl. Collaborative Intrusion Detection in Smart

Energy Grids. In *International Symposium for ICS & SCADA Cyber Security*, number 2010, pages 97–100. Electronic Workshops in Computing (eWiC), 2013.

[168] Emmanouil Vasilomanolakis, Shankar Karuppayah, Mathias Fischer, Max Mühlhäuser, Mihai Plasoianu, Lars Pandikow, and Wulf Pfeiffer. This Network is Infected : HosTaGe - a Low-Interaction Honeypot for Mobile Devices. In *Security and privacy in smartphones & mobile devices*, pages 43–48. ACM, 2013. ISBN 9781450324915.

[169] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. HosTaGe: a Mobile Honeypot for Collaborative Defense. In *7th International Conference on Security of Information and Networks (SIN)*, page 330. ACM, 2014.

[170] Emmanouil Vasilomanolakis, Jörg Daubert, Manisha Luthra, Vangelis Gazis, Alex Wiesmaier, and Panayotis Kikiras. On the Security and Privacy of Internet of Things Architectures and Systems. In *European Symposium on Research in Computer Security, International Workshop on Secure Internet of Things*. IEEE, 2015.

[171] Emmanouil Vasilomanolakis, Shankar Karuppayah, Panayotis Kikiras, and Max Mühlhäuser. A honeypot-driven cyber incident monitor: lessons learned and steps ahead. In *International Conference on Security of Information and Networks*. ACM, 2015.

[172] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys*, 47 (4):33, 2015. doi: 10.1145/2716260.

[173] Emmanouil Vasilomanolakis, Matthias Krügl, Carlos Garcia Cordero, Max Mühlhäuser, and Mathias Fischer. SkipMon: A Locality-Aware Collaborative Intrusion Detection System. In *International Performance Computing and Communications Conference*. IEEE, 2015.

[174] Emmanouil Vasilomanolakis, Shreyas Srinivasa, and Max Mühlhäuser. Did you really hack a nuclear power plant? An industrial control mobile honeypot. In *Conference on Communications and Network Security (CNS)*, pages 729–730. IEEE, 2015. ISBN 9781467378765.

[175] Emmanouil Vasilomanolakis, Michael Stahn, Carlos Garcia, and Max Muhlhauser. Probe-response attacks on collaborative intrusion detection systems : effectiveness and countermeasures. In *Conference on Communications and Network Security (CNS)*, pages 699–700. IEEE, 2015. ISBN 9781467378765.

[176] Emmanouil Vasilomanolakis, Carlos Garcia Cordero, Nikolay Milanov, and Max Mühlhäuser. Towards the creation of synthetic, yet realistic, intrusion detection datasets. In *IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies (DISSECT)*. IEEE, 2016.

[177] Emmanouil Vasilomanolakis, Shreyas Srinivasa, Carlos Garcia Cordero, and Max Mühlhäuser. Multi-stage Attack Detection and Signature Generation with ICS Honeypots. In *IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies (DISSECT)*. IEEE, 2016.

[178] Emmanouil Vasilomanolakis, Michael Stahn, Carlos Garcia, and Max Muhlhauser. On Probe-Response Attacks in Collaborative Intrusion Detection Systems. In *Conference on Communications and Network Security (CNS)*. IEEE, 2016.

[179] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. *Journal of Network and Computer Applications*, 32(5):1106–1123, sep 2009. ISSN 10848045.

[180] Nikos Virvilis and Dimitris Gritzalis. The Big Four - What We Did Wrong in Advanced Persistent Threat Detection? In *International Conference on Availability, Reliability and Security*, pages 248–254. IEEE, 2013. ISBN 978-0-7695-5008-4. doi: 10.1109/ARES.2013.32.

[181] Vivek Vishnumurthy and Paul Francis. On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks. In *International Conference on Computer Communications (INFOCOMM)*, pages 1–12. IEEE, 2006. ISBN 1-4244-0221-2.

[182] Vasileios Vlachos, Stephanos Androutsellis-Theotokis, and Diomidis Spinellis. Security Applications of Peer-to-Peer Networks. *Computer Networks*, 45(2):195–205, 2004.

[183] Miroslav Voznak, Jakub Safarik, and Filip Rezac. Threat Prevention and Intrusion Detection in VoIP Infrastructures. *International Journal of Mathematics and Computers in Simulation*, 7(1), 2013.

[184] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *ACM conference on Computer and communications security - CCS '02*, pages 255 – 264, New York, New York, USA, 2002. ACM Press. ISBN 1581136129.

[185] Matthias Wählisch, Sebastian Trapp, Christian Keil, Jochen Schönfelder, Thomas C Schmidt, and Jochen Schiller. First Insights from a Mobile Honeypot. In *ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 305–306. ACM, 2012. ISBN 9781450314190.

[186] Matthias Wählisch, Thomas C Schmidt, Andre Vorbach, Christian Keil, Jochen Schonfelder, and Jochen Schiller. Design, Implementation, and Operation of a Mobile Honeypot. Technical report, 2013.

[187] Ke Wang and SJ Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, pages 203–222. Springer Berlin, 2004.

[188] Ke Wang, G Cretu, and SJ Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advances in Intrusion Detection*, pages 227–246. Springer Berlin, 2006.

[189] Z Yang. PowerTutor: A Power Monitor for Android-Based Mobile Platforms. Technical report, University of Michigan, 2012.

[190] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the domino overlay system. In *Network and Distributed System Security (NDSS)*, 2004.

[191] Sebastian Zander, Grenville J. Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys*, 9:44–57, 2007.

[192] Zheng Zhang, Jun Li, C N Manikopoulos, Jay Jorgenson, and Jose Ucles. HIDE : a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *IEEE Workshop on Information Assurance and Security*, pages 85–90. IEEE, 2001. ISBN 0780398149.

[193] Chenfeng Vincent Zhou and Christopher Leckie. Relieving hot spots in collaborative intrusion detection systems during worm outbreaks. In *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, pages 49–56. IEEE, 2008. ISBN 978-1-4244-2065-0.

[194] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. A peer-to-peer collaborative intrusion detection system. In *International Conference on Networks*, pages 118–123. IEEE, 2005. ISBN 1424400007.

[195] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. Evaluation of a Decentralized Architecture for

Large Scale Collaborative Intrusion Detection. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 80–89. IEEE, 2007. ISBN 1424407990.

[196] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. *Computers & Security*, 29(1):124–140, feb 2010. ISSN 01674048.

[197] Zhi-Hua Zhou. When semi-supervised learning meets ensemble learning. *Frontiers of Electrical and Electronic Engineering in China*, 6(1):6–16, jan 2011. ISSN 1673-3460.

[198] Kiyana Zolfaghar and Shahriar Mohammadi. Securing Bluetooth-based payment system using honeypot. In *International Conference on Innovations in Information Technology (IIT)*, pages 21–25, dec 2009. ISBN 978-1-4244-5698-7.

COLOPHON

This dissertation was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*.

DECLARATION

I hereby confirm that the submitted thesis with the title "*On Collaborative Intrusion Detection*" has been done independently and without use of others than the indicated aids. I assure that I have not previously or concurrently applied for the opening of a promotion procedure with the doctoral thesis submitted here.

Darmstadt, May 2015

Emmanouil Vasilomanolakis,
July 26, 2016