Fraunhofer
AISEC

FRAUNHOFER-EINRICHTUNG FÜR ANGEWANDTE UND INTEGRIERTE SICHERHEIT AISEC

Hrsg.: Fraunhofer AISEC, Garching
Markus Zeilinger, Peter Schoo, Eckehard Hermann

Advances in IT Early Warning

FRAUNHOFER VERLAG

Markus Zeilinger, Peter Schoo, Eckehard Hermann

Advances in IT Early Warning

Fraunhofer Verlag

Kontaktadresse:
Fraunhofer AISEC Research Institution for Applied and Integrated Security
Parkring 4
D-85748 Garching near München (Germany)
Telefon +49 (0)89 3229986-292
Telefax +49 (0)89 3229986-299
URL http://www.aisec.fraunhofer.de/

Bibliografische Information der Deutschen Nationalbibliothek Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de abrufbar.
ISBN 978-3-8396-0474-8

Druck und Weiterverarbeitung:
IRB Mediendienstleistungen
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Bildnachweis Umschlag: © yanikap / Fotolia.com

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 800469, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon +49(0)711 970-2500
Telefax +49(0)711 970-2508
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Preface

For the last fifteen years, the Internet has been the technological backbone of our society. But as in many other cases throughout history, technology has been misused. Cybercrime, Cyberespionage, and Cyberactivism are the dark side of the coin.

We need preventive measures and global monitoring technologies to improve the resilience of the Internet against technical failure, natural catastrophes and criminal abuse.

In our daily life we have a well-functioning weather forecast system where public and private sensor nets give us significant knowledge about upcoming hurricanes or the weather for the next holiday trip. A similar system with the same accuracy would help us to mitigate the emerging threat situation significantly.

Today's signature-based and anomaly detection systems lack the capability to look ahead. We need systems that can inform us far in advance on technical malfunctions or threats to be expected. We need a clear and reliable model that effectively represents reality and allows predictions. This must also take into account economical constraints and result in an acceptable risk level.

This book gives a contemporary, state of the art overview on IT Early Warning Systems (EWS). It covers recent research of top scientist and institutions. The different articles present an overview of the current research and running projects like IP darkspace analysis, malicious botnet activity by flow data collection in large scale networks, integrated honeypot based Malware collection and analysis, signature based approaches or even hardware Trojans to name a few. All these approaches concentrate on single aspects of EWS, in a global IT EWS a combination is needed. We are still far away from systems which detect and react in a satisfactory way. Although nobody has found the silver bullet of early warning there are promising approaches described here. Further research, prototyping and implementation in collaboration with the ISPs and owners of the internet infrastructure will be needed.

The papers presented in this book are not only of academic interest for national security agencies but also create the possibility for new security products which can be used in the IT-departments of private companies. This kind of applied research creates new business models and improves the competitiveness of our industry.

Enjoy reading it!

Prof. Dr. Udo Helmbrecht
Executive Director of ENISA
European Network and Information Security Agency

October 2012

Workshop Organization

The workshop on Early Warning Systems – the third in row – has been organized by OÖFH Hagenberg, Linz, Austria and Fraunhofer AISEC Research Institution for Applied and Integrated Security, Garching near Munich, Germany.

Commitees

Program: Eckehard Hermann,
University of Applied Sciences Upper Austria
Peter Schoo,
Fraunhofer AISEC
Markus Zeilinger,
University of Applied Sciences Upper Austria

Local Organizer: Manfred Schleinzer,
Bundesministerium für Landesverteidigung und Sport

Sponsoring Institutions

Nokia Siemens Networks Management International GmbH for sharing the publication effort. Institute of Software Technology and Interactive Systems and SBA Research, Vienna University of Technology, as Organizer of the ARES Conference 2011, Vienna.

Notes from the Editors

Luckily and with the help of many other authors we managed to create a compendium type of collection of recent research results that are truly encompassed in the application domain of IT early warning systems. These contributions range from malware collection, detection and analysis on hosts and in flows, via artificial intelligence based approaches, up to economical aspects. In their majority they contribute to the sensor field of early waring. We had the pleasure to welcome almost all of the authors in the series of workshops that we organized between 2009 and 2011 in Munich, Germany, Linz, Austria and Vienna, Austria.

However, rather than publishing proceedings, which would be based on the contribution to the recent Vienna workshop in 2011 only, we have decided to address topics of research in the application domain of IT early warning in a wider format. We have hence invited a couple of other authors to contribute to this book to achieve a representative coverage of the research field. In this way we have managed to arrange a book with some added value, as we think. We are sure that the addressed topics have for the next decade – if not longer – a significant relevance in the research community and for early adopters on the practitioner side.

The publications of our Linz Workshop in 2010 have been done in the Journal "Datenschutz und Datensicherheit – DuD, 2011, Volume 35, Number 4" and can also be found under SpringerLink.

We like to thank Prof. Dr. Claudia Eckert and Prof. DI. Robert Kolmhofer for their support by the organization of our workshops during the last years and we like thank Prof. Dr. Udo Helmbrecht for his openness to enrich this book with his preface.

September 2012

M. Zeilinger
P. Schoo
E. Hermann

Table of Contents

The Many Facets of IT Early Warning – Open Issues, Current Research

Markus Zeilinger[1] and Peter Schoo[2] and Eckehard Hermann[1]

[1] University of Applied Sciences Upper Austria
Softwarepark 11, 4230 Hagenberg, Austria,
{firstname.lastname}@fh-hagenberg.at
http://www.fh-ooe.at
[2] Fraunhofer AISEC
Parkring 4, 85748 Garching (near Munich), Germany
peter.schoo@aisec.fraunhofer.de,
http://www.aisec.fraunhofer.de

Abstract. Systems encompassing functionality to detect anomalies or exceptional situations and that derive predictive conclusions out of observations made on traffic flows, in the infrastructure elements, on hosts or end user devices are called IT Early Warning Systems (EWSs). In this contribution the state of the art, open issues and research directions field of current IT Early Warning research activities are discussed.

1 Introduction

In securing cyber space the first choice is very obviously to take preventive measures such that operational risks, with economically viable effort, are minimized to an acceptable level. While this is first choice for preventing loss of integrity, control or information, cyber space is additionally protected by systems that detect anomalies or exceptional situations – systems that derive predictive conclusions out of observations made on traffic flows, in the infrastructure elements, on hosts or end user devices. These systems are EWSs, which enable pro-active security measures and may support to improve situational awareness or even suggest action in exceptional situations.

An EWS may encompasses both, signature based as well as anomaly detection systems. Certainly, anomaly detection makes early warning because of the capability to understand exceptional situation that have not been seen before or the detection of zero day exploits. This kind of early warning is oriented towards new threats. However, also signature based approaches take a role in EWSs. The recognition of flaws and attacks that have been detected earlier can be helpful information for the situational awareness of others, for example, in the case of collaboration with other parties. More strictly seen, collaborative warnings are less depended on anomaly detection and more dependent on sharing information. Hence, the more an EWS is focussed on anomaly detection and the more willingness exist to share information, the more effectively can such a system be operated, such that the frequency of

false positives alarms is minimized, while true positives for all participating and dependent parties are maximized.

Like in other application domains different from information technologies, that have been researched successfully for early warnings, the benefit is in helping to prevent loss or damage based on observations that allow helpful prediction. In a number of other application domains early warnings have shown to make useful contributions, if not life preserving warnings. Very relevant are, for example, tsunami, earthquake or rockfall and landslide early warning systems that are installed and do very useful jobs. Earthquake early warning is an illustrative example, because it shows that the early warning builds of the models of those geophysical processes that cause destruction. The propagation of primary and secondary seismic waves (*p-waves* and *s-waves*), can actually be modeled and is used to determine events, their geographical locations, travel time of shock waves and severity of an earthquake. In other words, there is a clear and reliable model that effectively represents reality and allows predictions. One recent example is the Japanese disaster preparedness measure for which mobile operators distribute warnings based on detected immanent seismic activities to users in areas the activities take effect.

In IT Early Warning (EW) we are far away from comparable processes, for example, considering only warnings to be issued to a subset of users that use a specific type of affected computer systems. When looking to the initial triggers that give the reasons to generate warnings, then the detection of new and upcoming exploits is less predictable in computer and communication systems than, for example, underlying models in geophysics on earth. One reason for this is evolution and innovation. Although we do, at this point in time, not have a model for IT EWS processes, this is no indication that EWS can not be effective in the future nor that their deployment can not be helpful for the protection of the cyber space. Successful IT EW processes are not only dependent on technology. Many of the instruments that can help improving the protection of the cyber space do yet need organizational, regulatory or legislation preparations.

In a comparative study of cyber attacks [1], it shows, for example, that spending the effort on ratifying Conventions on Cybercrime, installing sensors to make observations or deploying measures to mitigate cyber attack, it will turn out to become a positive effect for those nations that are acting pro-active in this field. The positive effect is that such nations are less often the places from where attacks are started and, due to higher awareness and protections, these countries are less often the targeted victims. Whereas not being pro-active increases the probability of becoming a victim or the base camp from where cyber attacks are started. This study indicates that positive effects can be caused when being pro-active in the protection of the cyber space. But it also seems to indicate that more research and technology development is required in EWSs at the same time, which improve tech-

nology and, additionally, positioning such systems under organizational, regulatory or legislation constraints into their ecosystems.

2 Buidling Blocks of EWS

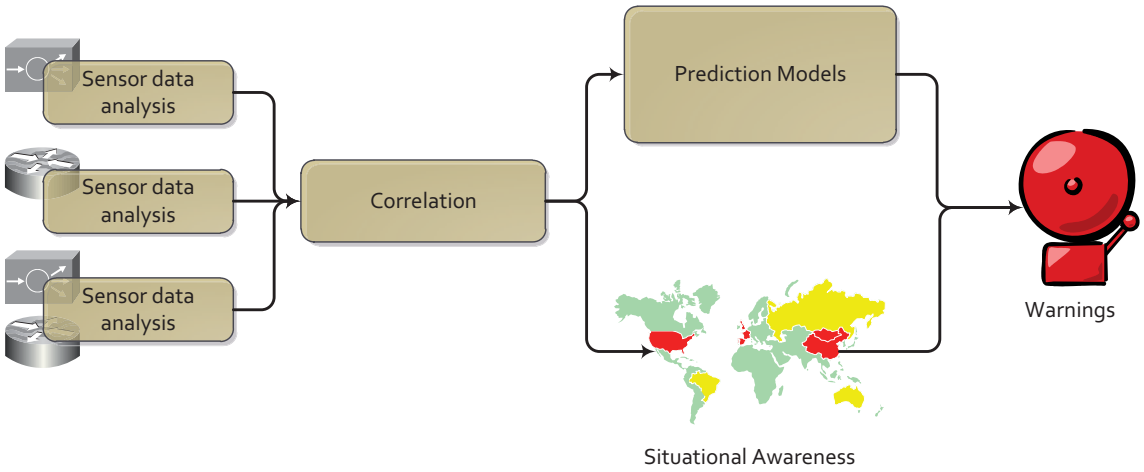


Fig. 1. Main building blocks of IT EW

A main building block of every EWS are sensors. Sensors mainly differ in the way they are placed in networks, the type of data they collect, and the methods they use for data analysis. For global EWS placement of a certainly huge amount of different sensors is necessary. It is critical to find the right placements for the sensors. These placements depend on the structure of networks and how network data is routed through them. Organizations like the Cooperative Association for Internet Data Analysis (CAIDA)[3] help to better understand the structure of the Internet and hence derive the right spots for sensor placement.

Sensors can collect network data on different levels of detail. Depending on the level of detail, analysis methods will be able to detect different kinds of anomalous activities and attacks. Collecting flow data, e.g. source and destination information, timestamp, transport protocol, byte/packet count of active connections, allows detection of network based attacks like probing, worm propagation activities or (distributed) denial of service (DoS) attacks. To be able to detect attacks on application level, sensors need to collect the whole payloads of packets, e.g. in `pcap` format. One further type of sensor that is widely employed in EW are honeypots.

[3] http://www.caida.org

Honeypots are mainly used to collect samples of (new) malware, which are then analyzed in (semi)-automatic analysis environments, e.g. Anubis[4]

In the past years many activies in EW focused on developing new analysis methods for network data especially in the area of anomaly detection. Surprisingly anomly detectors are only rarely employed in "real world" settings. Current research [2,3] presents two main reasons for this situation. On one hand basic assumptions for anomaly detection [4] are maybe not accurate anymore, due to evolution of networks and network traffic. On the other hand anomaly detectors often lack of practical usability, e.g. due to high false positive rates (FPR) or too little interpretation of results (semantic gap). Therefore more focused research in this area is necessary.

Beside basic detection of known and unkown malicious activities with signature-based and anomaly detectors, correlating events over time and space is an important requirement for EWSs. Events at different times and/or different sensors may be in itself harmless but relating them together may reveal them as single steps of multi-stage attacks. Correlating events is especially necessary to create situational awareness of global Internet activity.

To get the "early" in Early Warning effective prediction models are needed that are able to derive out of sensor data analysis conclusions about the prospective development of a situation, e.g. emerging attacks. This is crucial for success, cause only if the EWS is able to predict the prospective development of a situation, it is able to warn people possibly affected by this development in time and thereby prevent or at least reduce damage. As stated above we do not have such prediction models for EW in IT right now. This is a problem that still needs to be solved.

Warning as many people possibly affected by a damaging event as possible, is the main function of every EWS. Important aspects of warnings are the content of the warning and the way the warning is distributed. The warning should be as precise and clear as possible. It should be distributed through one or more communication channels that are appropriate for the warning cause and accessible for the people targeted by the warning. For example, it is more effective using speakers and/or fire sirens in villages/cities nearby the sea for tsunami early warning than only sending e-mail messages.

For all components in EW cooperation between many different stakeholders is necessary, e.g. governments, providers, companies. Since the Internet is not bound to company or country borders but global by nature, things happening in one part of the Internet can quickly effect other parts. Therefore it is most important that the stakeholders have a method for sharing relevant information as, for example, sensor data, analysis data, results of correlations, warnings. The key requirement for such a method is preserving anonymity and privacy for all participants. Obviously, legal regulations will be necessary to force and regulate participation in such collaborative information exchange.

[4] http://anubis.iseclab.org/

3 Overview on Contribution

In this contribution, "The Many Facets of IT Early Warning – Open Issues, Current Research", the state of the art, open issues and research directions field of current IT Early Warning research activities are discussed – a field in move, not easy to delineated. Petersen and Pohlmann discuss in their article "*An ideal Internet Early Warning System*" a model for the early warning and it will be shown which components need to be protected and why an early warning can only work through collaborative approaches and a variety of systems.

For the network and traffic analysis area, Zseby gives in "*IP Darkspace Analysis*" a general overview on the use of IP darkspaces for early warning. The author shows what kind of analysis can be done with IP darkspace data and what kind of results can be achieved. The general use of flow data collection to monitor large scale networks for early warning is demonstrated in "*Flow Data Collection in Large Scale Networks*". The Celeda et al. show the advantages and disadvantages, possible data processing techniques and present two use cases for flow data collection in end user and transit networks. Drasar, Vykopal and Winter focus in "*Flow-based Brute-force Attack Detection*" on detecting brute force attacks in flow-based network data. They discuss five different detection techniques, their strengths and their drawbacks. To show the fragility of some methods, several evasion techniques are demonstrated.

Regarding malware and under the headline "*Malware in Hardware Infrastructure Components*", Krieg and Weippl present and survey and discuss various approaches to detect malware or trojans that affect hardware, which can become a serious issue for network elements in the infrastructures we are using in daily life. Martin Brunner, Christian M. Fuchs and Sascha Todt present in the article "*Integrated Honeypot based Malware Collection and Analysis*" a new approach for integrated honeypot based malware collection and analysis with the intention to cover the entire malware execution life-cycle and thus make advances for better control of high-interaction honeypots. Their approach is based on the assumption that the ability to track this entire life-cycle facilitates a better understanding of current and emerging malware.

In "*Predentifier: Detecting Botnet C&C Domains From Passive DNS Data*" Frosch et al. show how a statistical model built based on 14 features from passive DNS data and other sources like WHOIS can be used to decide whether a domain name is used for malicious activities (C&C botnet traffic) or not. Lampesberger et al. present in "*Statistical Modeling of Web Requests for Anomaly Detection in Web Applications*" a new way to analyze anomalies on higher layers that use http as the communication protocol between distribute applications. In this self-adjusting and learning approach the anomalies in inter-application communications are validated by statistical models. Their evaluation results show a high precision and accurateness. Under the headline "*Automatic Generation of Generalizing Behavioral*

Signatures" Apel and Meier discuss the generalization of behavioral signatures, considering the question, which subset of a polymorphic malware family one need to know to know them all?

Hans Hofinger and Sascha Todt present in their article "*A Concept for Secure and Privacy-Preserving Collaborative Information Sharing*" a concept for secure and privacy-preserving information sharing across borders of administrative domains and show how existing techniques can be used to fulfill the manifold requirements of IT Early Warning.

Birkholz presents in "*IO: Deploying An Interconnected Asset Ontology To Enhance Information Retrieval Regarding Security Processes*" advances in the development of IO. He proposes a terminology that adapts definitions found in the context of organizational memory information systems. A generic concept of *Group* enables categorizing assets, enhancing information retrieval and new use cases demonstrate the interaction of IO with producers and consumers of information in the context of information retrieval.

Heiko Kirsch and Michael Hoche focus in the Section "*Between Early Warning and Emergency Response – An economical perspective*" on decision support for optimal service value and they discuss a concept to establish a mechanism that ensures proactive and reactive implementation of security countermeasures to meet real economic security needs.

4 Research Challenges

Analyzing past developments in EWSs we identify the following important aspects for future research: sensor data analysis and here especially anomaly detection, correlation of event data, tools for situation awareness presentation and methods for collaboration. We think that strong cooperative efforts from many different stakeholders are necessary to master these research challenges.

As stated above there has been a lot of research on anomaly detection in the past decade but many solutions lack of practical usability. This has to be address in future research urgently. Sommer et al. [3] describe key factors in developing accurate and usable anomaly detectors:

- understanding the threats an anomaly detector should be able to catch,
- keeping the scope as narrow as possible,
- reducing costs by reducing false positive rates (Axelsson claims in [5] false positive rates less than 10^{-5}),
- closing the semantic gap (e.g. by doing root cause analysis to better understand the mechanisms and organizations/people behind attacks) and
- sound evaluation on real world network traffic.

Bringing together results of many different sensors by correlating their events is another open research topic. Correlating events from different types of sensors,

happened at different times and at different spots in the Internet is a difficult task and we need expertise form the fields of data mining and knowledge engineering.

Last but not least collaboration in EWSs needs massive research effort. Collaboration is necessary in all aspects of EW, as, for example, exchanging sensor data, correlation of sensor events, warning distribution, regulations, etc. and therefore concepts and methods for information sharing are needed. These concepts and methods should allow the stakeholders to exchange information of any kind by simultaneously preserving anonymity and privacy for the participants. An essential question is, what can be achived by legal regulations and what can be solved by technical measures like privacy preserving techniques and cryptography?

There are currently some promising activities going on that try to address at least some of these research topics:

- WOMBAT[5] (**W**orldwide **O**bservatory of **M**alicious **B**ehaviors and **A**ttack **T**hreats): The WOMBAT project aims at providing new means to understand the existing and emerging threats that are targeting the Internet economy and the net citizens. To reach this goal, the proposal includes three key workpackages: (i) real time gathering of a diverse set of security related raw data, (ii) enrichment of this input by means of various analysis techniques, and (iii) root cause identification and understanding of the phenomena under scrutiny. The acquired knowledge will be shared with all interested security actors (ISPs, CERTs, security vendors, etc.), enabling them to make sound security investment decisions and to focus on the most dangerous activities first.
- ASMONIA[6] (**A**ttack analysis and **S**ecurity concepts for **MO**bile **N**etwork infrastructures, supported by collaborative **I**nformation exch**A**nge): The goal of ASMONIA is the development of a holistic security concept for mobile network infrastructures that satisfies the diverse requirements of modern networks. Integrity protection and attack detection solutions that exploit characteristics of resilient and flexible systems like cloud computing will therefore be integrated. The additional integration of collaborative information exchange mechanisms will improve the security level of modern communication networks. The mains tasks of the ASMONIA project are: Risk analysis of mobile networks and end devices, defining holistic and collaborative protection concepts, protecting the integrity of network devices, incorporation of resilient and flexible systems as essential protection components and development of attack detection and assessment techniques.
- NISHA[7] (**N**etwork for **I**nformation **S**haring and **A**lerting): The objective of Network for Information Sharing and Alerting (NISHA) is to further develop an existing prototype for an European Information Sharing and Alert System

[5] http://wombat-project.eu/
[6] http://www.asmonia.de/
[7] http://fisha-project.eu/

achieved under the FISHA project into a pilot version of the system. The expected outcome is a pilot network consisting of four local portals which are to be set up in member states, locally reaching citizens and SMEs. The network will function based on an organisational model proposed within the project frames. The project will include a study of organizational and legal aspects concerning functioning of the system as well as technical development and implementation encountered while establishing the pilot, with a focus on lessons learned and suggestions for improvement.

- FIDeS[8] (Early Warning and Intrusion Detection System based on combined methods of Artifical Intelligence): Aim of the FIDes project ist the development of a system for assisting in early detection of Internet based attacks. FIDes focuses on using methods of AI for assisting the operator in analyzing result of anomaly detection. This is done e.g. by giving feasable reasons for attacks, giving in depth knowledge on attacks, predicting further attack development and assistance in selecting appropriate countermeasures.
- PREDICT[9] (**P**rotected **RE**pository for the **D**efense of **I**nfrastructure Against **C**yber **T**hreats): PREDICT is a community of producers of security-relevant network operations data and researchers in networking and information security. Through its repository, it provides developers and evaluators with regularly updated network operations data relevant to cyber defense technology development.

References

1. S. H. Kim, Q.-H. Wang, and J. B. Ullrich. A comparative study of cyberattacks. *Commun. ACM*, 55(3):66–73, Mar. 2012.
2. C. Gates, C. Taylor. Challenging the anomaly detection paradigm: a provocative discussion. *Proceedings of the 2006 Workshop on new security paradigms NSPW 2006*, 21–29, 2007.
3. R. Sommer, V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security and Privacy*, 305–316, 2010.
4. D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222-232, 1987.
5. S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. *Proceedings of the 6th ACM conference on Computer and Communications Security (CSS'99)*, New York, USA, 1999, pp. 1-7.

[8] http://www.fides-security.org/
[9] https://www.predict.org/

An Ideal Internet Early Warning System

Dominique Petersen and Norbert Pohlmann

Institute for Internet Security
University of Applied Sciences Gelsenkirchen
{petersen,pohlmann}@internet-sicherheit.de

Abstract. Today, many manufacturers advertise that they have developed the one, ultimate Internet Early Warning System (IEWS) which shall protect the customers from all the dangers on the Internet. But what does an ideal and global Internet Early Warning System need in reality? In this article, a model for the early warning is constructed and it will be shown which components need to be protected and why an early warning can only work through collaborative approaches and a variety of systems. Then some of the current research work done by the Institute for Internet Security are outlined in the area of Internet Early Warning.

1 Introduction

With the theme "Internet Early Warning Systems" in the same breath the statements are often that the critical infrastructure protection (CIP) is very necessary and a primary goal of the state is to enforce this with all technological means. In addition to the transport-, energy-, financial- or service-area also the Internet - as we know it today - is a critical infrastructure. According to the assessment of the state this critical infrastructure must be protected.

To achieve this goal it is necessary to protect the functionality of the Internet and to keep it alive. As the Internet consists of many Autonomous Systems (AS), which are connected together, special attention must be paid to the protection of the individual sub-systems of this infrastructure [1].

1.1 Relevant aspects of the early warning

Here, two aspects are crucial. Firstly, it is important to identify threats as early as possible to reduce possible damage or at best even eliminate all harm. The containment and avoiding the damage here depends upon successful detection very much on the initiation of potential countermeasures. Secondly, it is necessary for the respective infrastructures to be adjusted and improved, so they are prepared for future requirements.

1.2 Threat scenarios

Currently, five different types of threats are distinguished and important on the Internet. In a DDoS (distributed denial of service) many requests and intense traffic

is generated from multiple distributed sources, so that the narrow-band target enters an overload situation and is not longer available for a normal use. Depending on how quickly the DDoS is done: in an emergency only seconds remain for an early warning.

Another threat are exploits, that exploit current vulnerabilities in software of operating systems. In networks where the exploits occur for the first time, an early warning of single network packets is almost impossible. Depending on the degree of diffusion speed and aggressiveness of the exploit yet unaffected infrastructures could get timely warnings. Again, the time for the early warning consists of only seconds in the worst.

Malware is currently one of the most invasive threats in the Internet. Systems infected with malware automatically try to attack other systems and redistribute themselves this way. How quickly this happens depends on the design of the malware. For an Early Warning System this means that the time for a warning can be a few minutes to several days.

An equally serious threat are botnets. Sometimes they consist of hundreds of thousands of systems controlled by a unnoticed botnet and execute commands, such as a DDoS. To identify which targets are attacked at which time, it is necessary to observe the communication of the botnet and to analyze it. The time for an early warning is here defined by the communication rate of the botnet and its structure and moves within a few minutes.

The last major threat scenario is made possible by the Internet routing. It has happened in the past that wrong distribution of IP prefixes were made in the routing, and so entire Autonomous Systems, and therefore also the entire traffic to these networks, diverted to other countries. Here, the early warning time also depends on the distribution speed of routing and is only a few minutes.

1.3 Response time for early warning

Considering the threat scenarios is clear that in general very little time for an early warning is available, and that all components involved must respond as quickly and efficiently as possible. In many cases it is impossible to issue a warning before the actual and concrete attack is launched. It will be easier however, if a warning of a potential threat is issued. Especially for a collaborative future Early Warning System infrastructures participators could be informed in time to avert possible damage.

2 Definition of an Internet warning system

Based on the basis of the goals, to improve the Internet and its infrastructures in terms of safety and reliability, and to generate a continuous situation overview of

the IT infrastructure, which is carried out in cooperation with public and private partners in the sense of a collaborative early warning, a definition for a Internet Early Warning System could be as follows: Based on reliable results and results from threats or in the event of IT security incidents that affect only few infrastructures yet, an IT situation overview is continuously updated, and when an adequate and relevant incident occurs, a qualified warning to potentially affected is disseminated in order to reduce the potential damage caused or avoided altogether.

2.1 Mandatory functional requirements

An Early Warning System therefore needs a set of functional requirements: The intrusion detection must be done at a time before concrete damage occurred and early enough to minimize potential damage. Here it is important to consider both, already known and unknown, attacks are detected.

The decision-making process and the development of countermeasures have to be supported. This can e.g. performed by analysis tools and results visualization. Expert systems will help in the decision making here. Ensuring and collection of evidence for forensics must be guaranteed in order to perform legal prosecutions later.

The current status and the development of the Internet traffic must be monitored constantly. Questions, how the infrastructure needs to be extended or which technologies increase or decrease of importance in the future, are important here.

The current IT situation overview along with an overview of all security events must be generated continuously. Here, appropriate visualization methods help for the situation analysis. Other requirements follow directly from the fact that the Early Warning System itself needs to be protected. The stability and the security of the system against attacks, maintaining the privacy, the maintainability and the performance all are relevant aspects.

2.2 Asymmetric threats

Another problem is the asymmetric threats. Many attacks are carried out globally and are not aimed at a specific location. The best example illustrates a distributed denial of service attack (DDoS).

The response to a security incident occurred is currently initiated only locally and therefore concerns only the particular infrastructure which acts against these threats. All victims of a global attack must therefore execute the same or similar reactions and counter-measures to reduce the damage or avert all harm. The total cost is thus multiplied almost to the number of victims and the relevant countermeasures.

Target of an Internet Early Warning System must therefore be to initiate effective responses to incidents for all partner systems, and this preferable in an automated manner.

3 Structure of an Internet Early Warning System

Based on architecture and requirements of the particular operating company a model for an Early Warning System can be set up (see figure 1). Foremost, the system is defined by the targets that are to be achieved.

Equally important are the legal settings and conditions, in which the whole early warning is performed. Depending on how the legal conditions in the state and the company look, the Early Warning System can have more or less restrictions. Relevant here are mainly the privacy standards that need to be adhered to, the protection of the trust and the respective contract law. Not infrequently, the legal framework defines the possibilities of an Early Warning System to achieve the required goals.

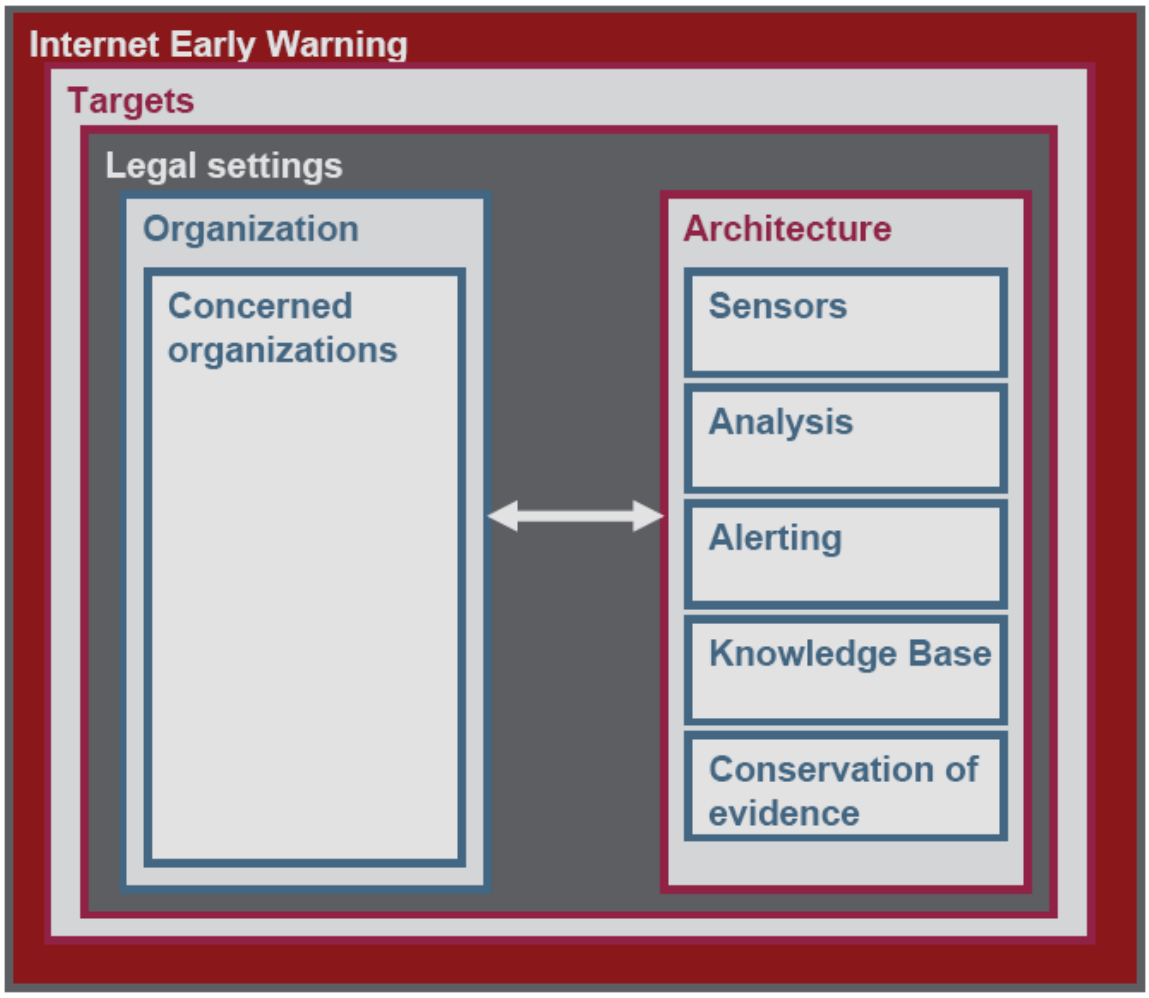


Fig. 1. Model of an Early Warning System

Afterwards the respective company or organization, which is operating the Early Warning System, and the partners involved (concerned organizations) determine the model. The partners here may assume two different roles, active and passive. In the active role the participants are involved in the establishment and operation of the Early Warning System, e.g. by providing sensors, operating a situation awareness room or by the creation and execution of countermeasures. Passive participants consume almost all the information provided by the other Early Warning Systems. Frequently, these are home users and small businesses.

The operator of an Early Warning System usually has a precise definition of the organizational units with their respective relationships and clearly defined responsibilities. In order to act more quickly, the necessary information flows and possible responses must be clearly agreed. It is important for such a company to have a very short decision process, efficient paths for the distribution of information as well as clearly defined responsibilities to warn and to react in an emergency pretty early.

3.1 Technical Implementation

The architecture inside of the early warning model represents the technical components that need to be implemented. The sensors, which are distributed to each strategic positions in the network to be monitored and set up, collect the data, which form the basis of the current status. The distribution of sensors depends on which parts of the network are critical how representative the index shall be. Several types of sensors have already been developed. It is possible to use sensors that create a map of the total traffic, such as flow data, packet-based statistics sensors, honeypots, log and availability data or approaches that capture all traffic.

Especially with the sensors used, it is extremely important to pay attention to both, the privacy protection and the preservation of evidence. This can be achieved by methods of pseudonomisation and anonymisation. Nevertheless, it must be ensured that for the particular operation area the ideal sensors are used. For example the DE-CIX, the world's largest commercial Internet exchange (CIX) point has a peak traffic of 1.8 Tbit/s. The analysis component of the sensors must therefore be very powerful, if not the included information will be strongly reduced. Alternatively, the use of sampling can be applied, wherein in a certain time or after a certain number of packets flown through the network only a sample is taken from and then evaluated.

3.2 Core of an Early Warning System

The analysis and recognition module (Analysis) is the core component of an Early Warning System. It is responsible for identifying security incidents and to transmit these in form of alarms. In order for the threat detection works, a number of technical components is necessary (see figure 2, [2]).

The measured data of the installed sensors are sent to the signal layer. The data is then filtered for relevance and analyzed. In addition, here is the detection of abnormal and safety incidents, such as with misuse detection and anomaly detection methods. These algorithms generate events that represent a particular occurrence and include information that contributed to the construction process.

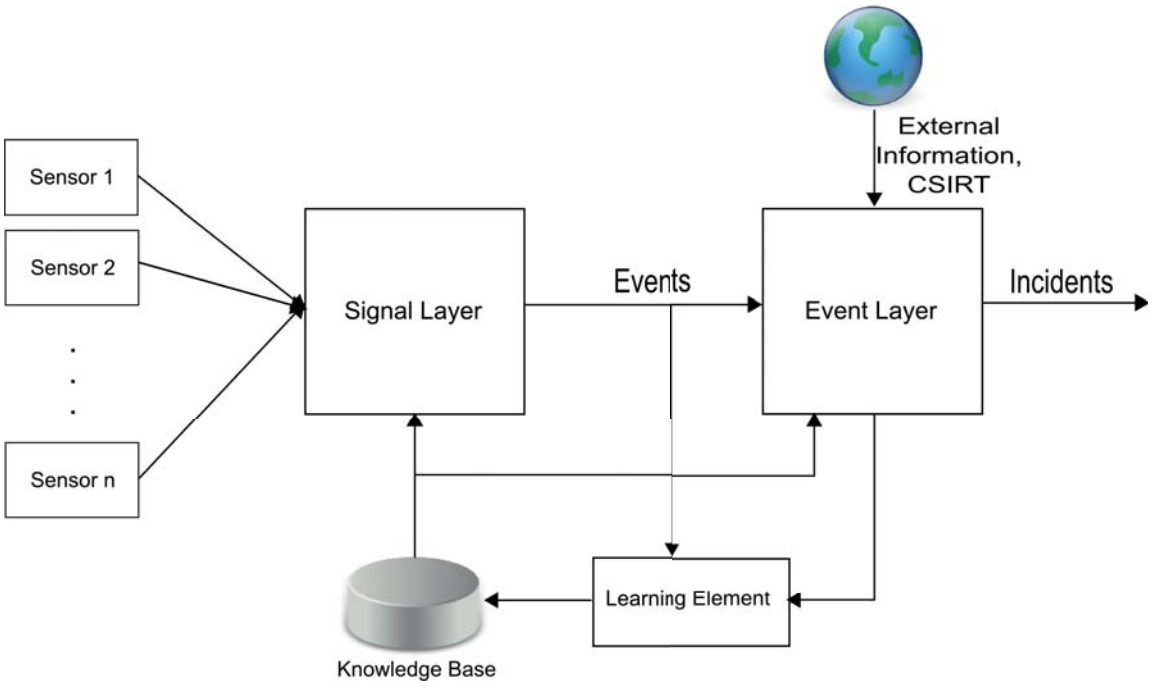


Fig. 2. Technical components of the detection methods

These events will now be passed to the event layer, where they are correlated. It is advisable to include more information about incidents or security flaws from external, non-technical sources (e.g. CERT's) in order to make a better conclusion. If the analysis concludes that the individual events are not only an anomaly, but a specific incident or particular threat, an alarm will be generated and forwarded to the responsible person in the company.

The biggest problem in identifying is firstly the vast amount of data that must be analyzed. On the other hand, the big challenge is to identify previously unknown attacks and slowly evolving trends.

To further improve the detection method, an Early Warning System has always a learning component (element). Using the information returns from the events and the results from the event layer, the algorithms are adjusted adaptively. For example, some algorithms that monitor network traffic have to be adjusted after a

new service has been added, which was not previously known. The results from the learning component flow as modeled findings in the knowledge base.

Another important aspect implements the knowledge base. It contains the knowledge about the environment in which the Early Warning System is used as well as information about the normal behavior of the network and attack signatures. Ideally, there are also concrete countermeasures regarding certain, already known incidents and practices when problems occur in the knowledge base. So they can really help and be supportive, the contained data must be always kept current. This can be achieved e.g. through the automatic generation of virus and attack signatures, updating the normal state of the network traffic or functioning processes to solve previously unknown problems.

The challenge with this is the continuous acquisition and storage of knowledge. As part of an expert system, the knowledge base does not only provide information for the solution of well-known problems which are available, but also provides intelligent support when editing unknown incidents in which they propose similar events and suppression instructions.

3.3 Legal consequences

If a concrete attack was detected at various levels of the recognition process and ideally successfully repelled, it is about to call the criminals legally and financially to account. To have a chance in the digital age at the court, the conservation of evidence must be carried out consistently and trustworthy.

All information about the attacker are important, his approach and the damage that the attack caused. Also these two important aspects are to be kept for further use. On the one hand, the privacy protection laws (policies) must be followed, i.e. the access to the stored data must be restricted and linked to a specific detected incident. This is to protect personal data against misuse. On the other hand, the authenticity of the evidence can be ensured by making tempering technically impossible.

3.4 Overall architecture of the Ideal Early Warning System

The previous explanations help to identify a number of important aspects that significantly determine the overall architecture (see figure 3).

For instance, even at the local systems the incident management and the countermeasures are defined by different rules. In addition to that, due to the different environments, not every countermeasure can be applied to any network. Moreover, in several countries, there are different legal frameworks which must be observed.

If a threat is detected in a situation awareness room, it is important for an early warning to communicate the threat description to all participants involved, and this preferably faster than the spread of the attack. Finally, it's the Early Warning System itself which has to be robust in order to perform its actual task.

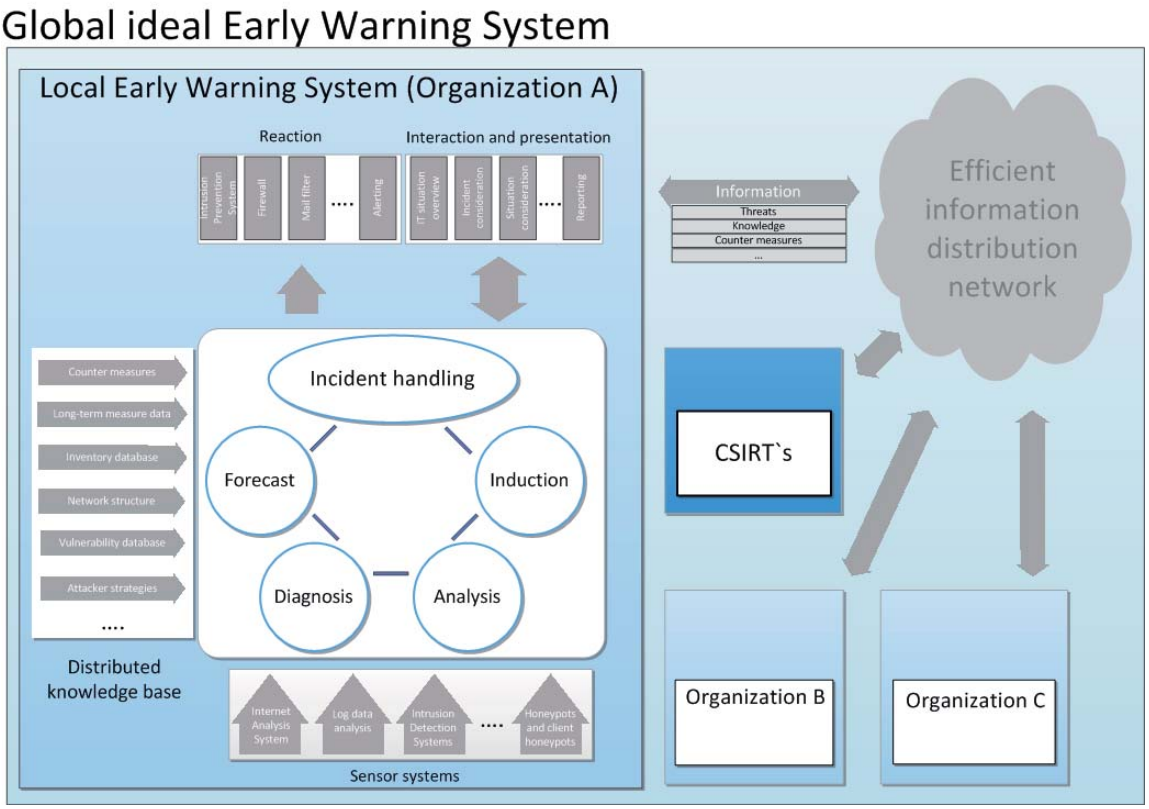


Fig. 3. Structure of a global, ideal Early Warning System

In order to respond to the threats today it is needed to act globally and collaboratively. Large enterprises and governments need to operate sensors throughout the Internet and analyze this data locally. If incidents are identified, that must be as soon as possible spread to both active and passive participants involved by an efficient information distribution network. Also, certain countermeasures must be carried out collectively within the early warning participant network.

Also important is a distributed knowledge base, in which already occurred incidents are stored and can be accessed by all partners for a quick and ideal response.

4 Developments of the Institute for Internet Security

The Institute for Internet Security - if (is) of the University of Applied Sciences Gelsenkirchen has been working for more than seven years of applied research of Internet Early Warning Systems. Within this project several technical implementations have been developed, of which three selected projects will be presented.

4.1 Internet Analysis System

Using the sensor-based Internet Analysis System (IAS), which was developed as a research and development project of the if(is) in collaboration with the Federal Office for Information Security (BSI), local and global overviews can be created and analyzed in order to generate early warnings ([3]). The Internet Analysis System provides the core component of the Internet Early Warning Systems of the if(is) ([4]).

Special key aspects of the project are a privacy-friendly collection of network information and optimizing the amount of information data to store long-term, thus to enable the analysis of trends and developments over long periods.

4.2 Goals and objectives of the IAS

The task of the Internet Analysis System is on the one hand the analysis of local communication data in defined subnets of the Internet (networks), and on the other hand the creation of a global view ([5]) on the subnets or the whole Internet with the help of combining the many local views by the network operator ifself or an independent party, such as the Institute for Internet Security.

The functions of the Internet Analysis System can be divided into four sections: Build-up of the knowledge base, description of the current status, alerting, and forecasting.

The main task of building up the knowledge base is a comprehensive analysis and interpretation of the communication parameters of the Internet traffic with the goal of discovering technology trends, relationships and patterns that represent different states and views of the Internet. Based on this knowledge base anomalies are found with actual measurements and reasons for the state changes which are analyzed and interpreted. This happens, inter alia, with methods of artificial intelligence (AI) such as probabilistic neural networks (PNN), which can be implemented on graphic processing units (GPU) ([6]). It is important to find out whether the state anomalies are of natural origin, such as through a technology change, or whether a malicious attack is responsible. If such a malicious attack is present, the patterns can be identified which characterize the attack, in order to detect these faster in the future.

By the exact knowledge of the current state of a communication line and the aid of historical, i.e. previously collected information from the knowledge base a warning message will be generated, if there are are significant changes in traffic or communications data. After that measures can be taken to protect and preserve the functionality of the Internet.

Another important function is to show the status with a visual representation of the state of the Internet, similar to a weather chart or traffic jam map. It is important to make urgent decisions - especially at risk - easier and faster than ever

in order to explain complex issues to a third party. Here, it is not only warned of dangers, but also the positive situation when the monitored networks are alright is illustrated. In addition to a simplified view of complex structures the visualization system must be supportive in order to signal anomalies, such as spam attacks or malicious malware attacks, early enough so that preventative arrangements can be taken and risks can be minimized.

By studying and analyzing the collected data from the IAS, the technology trends, the relationships and the patterns, it is possible through a process of evolution of the results obtained, to make predictions about changes in state of the Internet (e.g. with the help of neural networks). In this way, attacks and major changes can be identified pretty early to forcast damage effects and capacity bottlenecks.

4.3 Functionality of the IAS

The Internet Analysis System consists of sonsors, which passively tap the network traffic of communication lines of different networks and count communication parameters on different levels of communication (OSI model). In an evaluation system the communication parameters are evaluated from different perspectives and displayed well-arranged.

In order to deliver results for a meaningful situation overview, the Internet Analysis System requires a very large amount of raw data, i.e. many counters of different communication parameters on all communication levels (OSI layers 2 to 7). All analyses, which the evaluation system performs, are based upon these raw data.

The sensors can send the raw data to one or more evaluation systems. Each organization is able to monitor their communication with the Internet and perform their own analyses with its evaluation system. To achieve a global and representative view of the Internet, sensors have to be placed in different types of networks, such as Global Tier One providers, transit providers, Eyeball Internet Service Providers (DSL providers), content providers and large enterprise networks, as well as in different regions.

4.4 FIDeS as intelligent correlator

The Institute for Internet Security was involved in another research project called FIDeS (Early Warning and Intrusion Detection System on the basis of combined methods of artificial intelligence), which is primarily a smart event correlation.

Intrusion Detection Systems (IDS) are widely used to protect corporate networks. Because they detect attacks against computer systems, among other things, they help to discover the theft of important corporate know-how and to stop this. These systems, however, currently still suffer from two major problems: First, they work mostly signature-based and can therefore only detect attacks that are already

known and for which a signature exists. The second problem: Due to the high false positive rate and the mass of the resulting events provided by these systems, security managers are not able to process all events sufficiently.

In order to identify the attackers, a huge and ever-growing expertise is needed. Only in this way related scenarios can be taken to events together and correlated with other information such as vulnerability or inventory databases. FIDeS want to solve these two problem points based on past research projects and new ideas, to help security managers in their daily work. The aim is to improve the intrusion detection and the subsequent forensic analysis. If possible even predictions shall be taken to prevent critical incidents from the outset. Using these results, information can then be analyzed across the enterprise in order to detect attack scenarios, that may not have been noticed in one location, but maybe on a larger scale.

4.5 Technical implementation of FIDeS

FIDeS uses as many well established and standardized open-source systems and open formats for data exchange as possible. Thus, further sensors or other components connected to the base architecture can be implemented with any programming language. The core sensor used here is the privacy-compliant Internet Analysis System.

The focus at FIDeS is the user who should to be supported in his daily work, the monitoring of the security situation. For this reason, special attention is given to the user interface. Large amounts of information can thus be detected quickly and intuitively, in order to take decisions for actions in time.

Current technologies in the field of Web 2.0 help for this purpose in making the system easy to handle and well configurable.

4.6 iAID with flow-based detection for very high Internet connectivity nodes

In the research project iAID (innovative Anomaly- and Intrusion Detection) an innovative anomaly detection and a new generation of IT Early Warning System shall be realised, which has a high recognition rate for threats, taking current privacy protection aspects into account, and shall be able to also analyze large quantities of data in real time. It will be investigated, how it can respond appropriately to threats using information fusion and creation of taxonomies of anomalies.

It must be developed a suitable way of collecting information about the network traffic. This is to fulfill the three properties: Highly detailed description of network traffic at all layers of the communication stack, resource-efficient collection and storage of useful meta-information from the communication data and observing and complying with privacy protection issues.

Once an optimal information collection system was implemented, in the next step a methodology for the evaluation and classification is developed. The focus here is to reduce the number of messages.

An important component of the new anomaly detection system (or Early Warning System) will be a feedback module, which will allow the analyst to review the decisions of the anomaly detection system and thus improve future decisions.

Thus only major anomalies to reach the administrator, an intelligent filter use the feedback of the analyst to filter out all flows or anomalies, which have similarity to those who were considered unimportant. The filter shall also use methods of artificial intelligence to perform this classification.

5 Summary

This article has presented the importance of Internet Early Warning regarding current threats and how much time there is to protect the infrastructure in case of an attack.

Furthermore, a definition of an Internet Early Warning System was described and demonstrated the functional requirements.

Based on the definition and requirements the structure and the technical realization of such an Internet Early Warning System have been carved out. Finally, it was shown what an ideal Internet Early Warning Systems should contain.

The article was rounded off with the presentation of three projects in the area of early warning, which the authors have realized in the recent years.

References

1. Sebastian Feld, Tim Perrei, Norbert Pohlmann and Matthias Schupp, *Objectives and Added Value of an Internet Key Figure System for Germany*. In Proceedings of the ISSE 2011 - Securing Electronic Business Processes - Highlights of the Information Security Solutions Europe 2011 Conference, 2011.
2. Sascha Bastke, Mathias Deml and Sebastian Schmidt, *Internet Early Warning Systems - Overview and Architecture*. In 1st European Workshop on Internet Early Warning and Network Intelligence (EWNI 2010), 2010.
3. Malte Hesse and Norbert Pohlmann, *Internet situation awareness*. In eCrime Researchers Summit, 2008, 1–9., 2008.
4. Dominique Petersen, Kilian Himmelsbach, Sascha Bastke and Norbert Pohlmann, *Measuring and warning*. In kes – Professional journal for information security, issue 5/2008, 2008.
5. Norbert Pohlmann and Marcus Proest, *Internet Early Warning System: The Global View*. In ISSE 2006 Securing Electronic Business Process, Vieweg, 2006.
6. Sascha Bastke, Mathias Deml and Sebastian Schmidt, *Combining statistical network data, probabilistic neural networks and the computational power of GPUs for anomaly detection in computer networks*. In Workshop Intelligent Security (SecArt 2009), 2009.

IP Darkspace Analysis

Tanja Zseby

Fraunhofer Institute FOKUS, Berlin, Germany
tanja.zseby@fokus.fraunhofer.de
http://www.fokus.fraunhofer.de

Abstract. An IP darkspace is a global routable IPv4 or IPv6 address space that contains no active hosts. All traffic sent to darkspace IP addresses is unsolicited, because it is destined to non-existing hosts. Since malware often uses random addresses, e.g., for network scanning or for address spoofing, a significant portion of malware traffic gets directed to darkspace. This traffic provides a useful information source to identify and analyze security incidents and misconfigurations.

1 IP Darkspace Traffic

Packets observed in IP darkspaces originate from misconfigurations or from processes that send data to randomly selected IP addresses. Most packets observed in darkspace have one of the following origins:

- **Scanning**: A large amount of traffic in darkspace originates from random scanning activities. This is caused by attack tools or worm spreading mechanisms that look for new victims by sending requests to random addresses or ports. Vertical scans target a specific IP address and multiple ports. Horizontal scans target multiple IP addresses and a specific port.
- **Probing**: Probing targets a specific IP address and a specific port. Probing traffic can originate from one or multiple source addresses. Probing sources sometimes use spoofed source addresses or are part of a coordinated attack with botnets.
- **Backscatter**: Malware often substitutes the real source IP address by randomly chosen IP addresses (address spoofing) in order to conceal the origin of an attack. Backscatter traffic is a side-effect of such attacks with spoofed addresses. When an attacker sends requests with spoofed source addresses to a victim, the attacked hosts respond by sending packets to the spoofed addresses. If the spoofed addresses are selected randomly they may include darkspace addresses and the response packets get directed to the darkspace. That means that backscatter traffic is sent to darkspace addresses by legitimate hosts that are under attack.
- **Misconfigurations**: Some packets end up in darkspace because IP addresses are wrongly configured. This can be caused by typos, by the use of example addresses from literature, or wrongly configured software. Traffic caused by misconfigurations is especially seen in darkspaces that contain simple addresses that

are more likely to be used as default values or as example values in texbooks (e.g. 1.1.1.1, 1.2.3.4). [22] describes the detection of several misconfigurations by analyzing traffic to specific darkspace addresses.

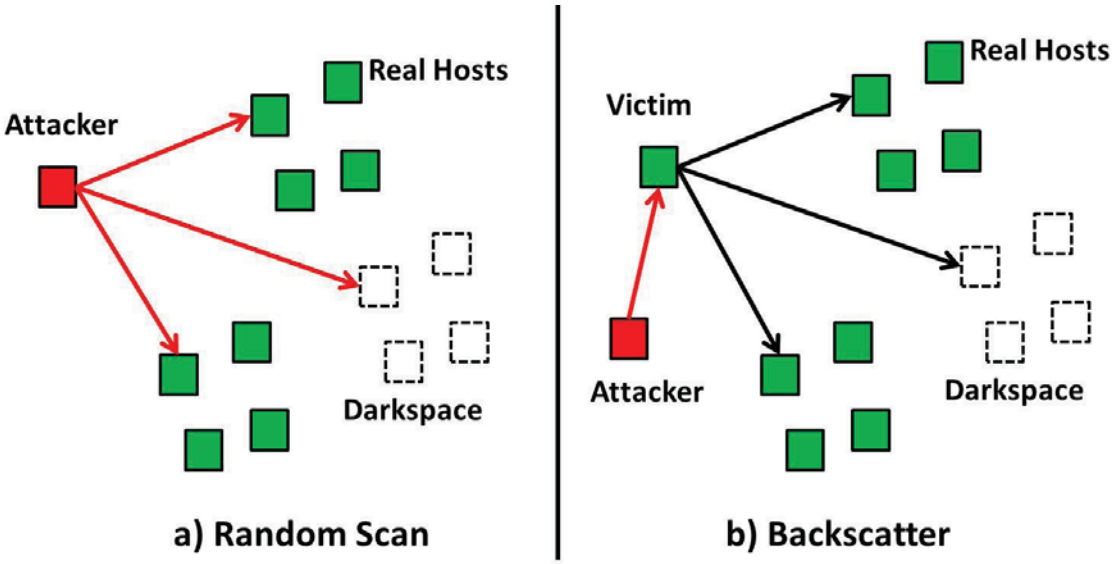


Fig. 1. Darkspace Traffic: a) Random Scan, b) Backscatter

2 Objectives of Darkspace Analysis

The analysis of darkspace traffic is helpful in many ways. We differentiate the following objectives for analyzing darkspace data:

- **Attack Detection and Malware Analysis** : Due to the use of random addresses in malware, a fraction of the attack traffic, caused by scanning or by backscatter, ends up in darkspace. Such traffic provides valuable hints about ongoing attacks and attack preparation activities. It can be used to assess the scale of incidents and helps to analyze malware behavior, such as scanning routines or malware signatures. Methods to use darkspace data for attack detection and the analysis of malware are described for instance in [18], [13], [1], [21], [2], [22].
- **Reducing Unwanted Traffic**: One purpose of darkspace analysis is to identify the origin of darkspace data and ask cooperative owners of infected hosts or malconfigered software to fix the problem. With this one can reduce unwanted traffic in the Internet and "'clean up"' a specific address space. A reduction

of unwanted traffic makes an address range more attractive for future use. [22] describes such cleaning efforts.
- **Network Analysis**: Darkspace traffic origins from all over the world. Events that impact Internet connectivity in specific locations (e.g., natural disasters, network failures, censorship) leave traces in darkspace traffic, because traffic from these locations is absent, reduced or altered. Darkspace analysis has recently been used to locate and investigate network outages caused by natural disasters, network failures or censorship [10].

Darkspace analysis helps in early warning efforts by reporting observed incidents. It helps to tune measurement efforts towards suspicious activities and to warn potential future victims. A pre-requisite for this is that the information from darkspace traffic is processed fast enough to detect incidents of interest. It is also useful to share darkspace data among different organizations to enable coordinated detection and warning efforts. For this it may be necessary to remove or anonymize sensitive information, such as the darkspace address range or IP addresses of infected hosts, beforehand.

3 Darkspace Analysis Methods

The amount of data received in a darkspace can be huge and it contains a lot of uninteresting and repeating events. It is crucial to extract the right information from the observed packets. We here describe different methods that have been proposed for the analysis of darkspace data and the detection of events of interest. Packets are usually first classified into subgroups. Then each subgroup is processed to extract the required information.

3.1 Packet Classification

Some basic analysis can be done by just observing the total packet count of the packets in darkspace. But more sophisticated analysis efforts require the classification of observed packets into subgroups. Classification requires the inspection of packet header fields, packet payload or packet arrival times. Most commonly used classification rules are based on IP addresses, protocol, port numbers and TCP flags.

Scanning software often uses TCP SYN packets but also UDP packets are used to scan the network. Typical backscatter packets are TCP SYN-ACKs or TCP RST packets that are sent as response to TCP-SYN packets. Also ICMP response packets as answer to ICMP echo requests and other packet types can be seen. [20] and [18] contain an overview of different backscatter packet types that can be observed in IP darkspaces.

Also the investigation of other fields make sense. The time-to-live (TTL) value provides hints about the operating systems that runs on the machine that sent

the packets [22]. For the investigation of network outages, it is useful to classify darkspace traffic based on the geolocation of hosts [10].

Building meaningful classes can result in quite complex rules. The iatmon tool introduced in [6] provides a special software for monitoring one-way traffic. It classifies darkspace traffic into 14 different source categories. It identifies backscatter traffic, vertical and horizontal scanning as well as probe traffic. The tool also analyzes packet inter-arrival times (IATs) to distinguish high-rate and low-rate sources and detect stealth scanning.

3.2 Packet and Byte Counts

The overall packet count is an indicator for increasing or decreasing overall traffic in the darkspace. It allows a first assessment whether something unusual is going on. A significant drop of the overall packet count can indicate an end of a large attack or severe network problems somewhere in the Internet.

Byte counts usually correlates with the packetcount. If byte counts increase while the packet count remains constant, packet sizes have increased. This may be caused by an increase of packets that contain malicious code in their payload. But also other reasons like misconfigurations can contribute to larger packetsizes. Overall packet and byte counts provide only very coarse grained information about changes in the darkspace traffic. More sophisticated analysis requires the classification of packets as described in section 3.1.

Packet and byte counts per traffic class reveal what kind of traffic increase. With this one can detect new malicious activities. An increase of the number of scanning packets to a specific port may reveal a new vulnerability. An increase of backscatter traffic can indicate that new DDoS attacks with randomly spoofed addresses are active.

3.3 Time Series Analysis

The detection of interesting activities in darkspace data requires the analysis of the temporal evolution of selected metrics. Different methods can be used from time series and trend analysis or change point detection. Time intervals for the analysis need to be chosen in accordance to the observed metric and expected dynamics.

The authors of [3] use the cumulated sum (CUSUM) algorithm to detect changes in statistical properties of the overall packet count from a darkspace monitor. They also propose to use a dynamic sliding window to automatically detect nested changes caused by overlapping anomalies. In [12] temporal and spatial correlations of time series of unwanted traffic are investigated. The authors look for long range dependencies in darkspace data and get different results for UDP and TCP darkspace traffic.

3.4 Feature Distributions

The investigation of feature distributions requires packet classification based on the feature of interest as shown in section 3.1 . The analysis of feature distributions can be combined with a filtering to narrow the scope of the analysis.

Darkspace traffic often originates from processes that use randomly chosen addresses and port numbers (e.g. scanning) or that target a specific address or ports (e.g. DDoS). Therefore IP address and port number distribution can reveal useful information. The increase of traffic to a specific port can provide information about a new vulnerability. The sudden occurrence of many new source IPs can be caused by a botnet or address spoofing.

One metric of interest that has been widely used in darkspace analysis is the number of unique source IP addresses. During the conficker outbreak in 2008 a sudden increase of unique source IPs could be observed that sent packets to port 445 [1].

The analysis of destination IP distributions from scanning traffic can also provide information about attack tools. It can be seen if those tools use random scanning within a subnet and whether random source ports are used for attack packets. Darkspace analysis reveal specific patterns of scanning software that is useful to identify those tools. For instance in [1] it is shown that only 1/4 of the addresses in the darkspace are scanned by conficker. This behavior is probably the result of a bug in the random number generation in the Conficker scanning software.

A nice summary metric to capture the concentration or dispersion in distributions is the sample entropy. Some large events in darkspace can be identified by just looking at the entropy instead of investigating the whole distribution [23].

Figure 2 shows the entropy analysis of data from a large /8 darkspace at the University of California, San Diego (UCSD). The graph shows data observed in February 2012. The diagrams show the total number of packets (lowest, chart 4), the number of backscatter packets in accordance to classification rules in the iatmon tool (chart 3), the entropy of the source and destination IP addresses H(sIP), H(dIP) (chart 1) and the source and destination port numbers H(sPort), H(dPort)(chart 2). It clearly can be seen how changes in backscatter data influence the sample entropy of address and port number distributions.

When the amount of backscatter traffic increases we see a significant drop in the source port entropy. That means that in the port distribution a specific port sticks out. This is the port that was used by the attacks that caused the backscatter data. We also see a decrease of H(sIP), due to specific hosts that are sending backscatter data. The destination port entropy H(dPort) increases, due to the use of random source ports by the attack tool.

The entropy H(dIP) of the destination addresses is already very high, because most of the dark addresses receive traffic. So we do not see any significant changes due to additional backscatter traffic.

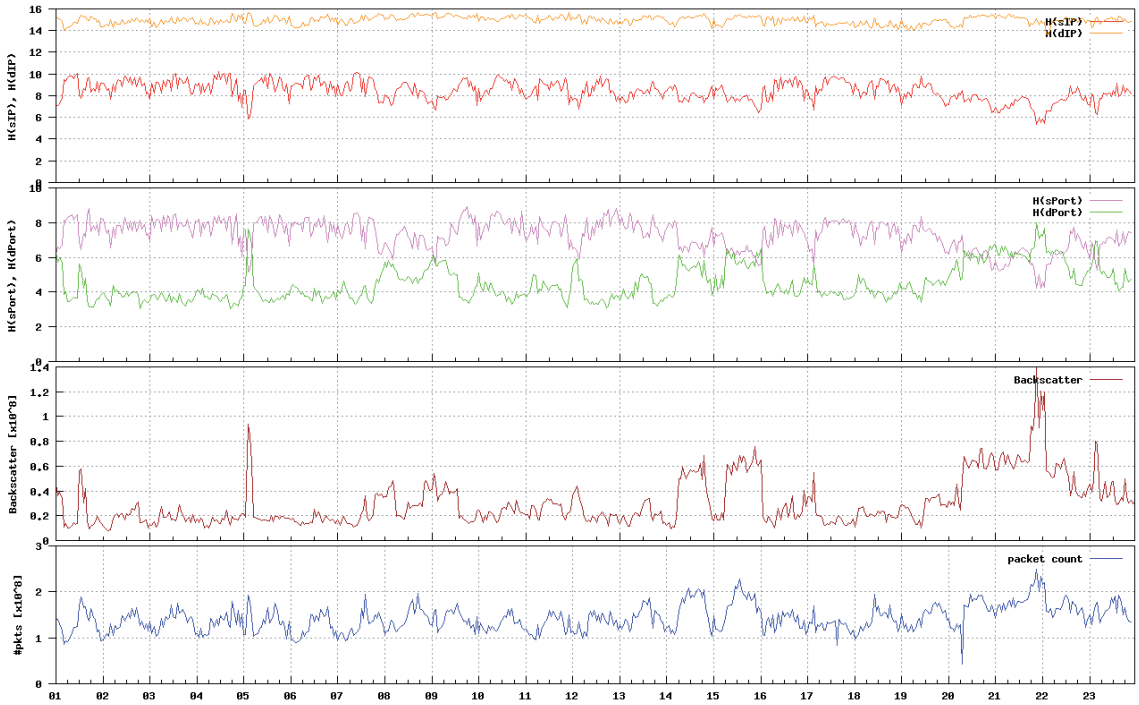


Fig. 2. *Entropy and Backscatter*

Besides addresses and port also the distribution of other metrics is of interest. In [6] the distribution of packet inter-arrival times per source group is used to identify stealth and slow attacks. In [22] the distribution of TTL values is used to investigate what operating systems contribute to darkspace traffic.

3.5 Further Techniques

Most techniques used for general network traffic analysis can also be applied to darkspace traffic. A detailed analysis is often done manually after some unusual patterns have been detected by automatic methods. Apart from time series analysis and distributions also the power spectrum of darkspace metrics has been investigated. The authors in [13] use power spectrum and detrended fluctuation analysis to look for long range dependencies. They also propose a 3D visualization of addresses and port numbers to assist in detecting unusual events. Also the combination of techniques makes sense. [19] describes the estimation of number and intensity of DoS attacks in the Internet based on the analysis of backscatter data. The authors group the observed packets into flows based on destination IP and protocol and then investigate TCP flags and ICMP payload per flow. They check if the assumption of randomly spoofed addresses holds by checking the uniformity of addresses distribution of observed packets using the Anderson Darling test statistic.

4 Darkspace Structures

Setting up a darkspace is not very difficult if an appropriate address space is available. [5] describes a darkspace setup and lessons learned from darkspace operation. But IPv4 addresses are a scarce resource. We expect that in future only few organizations can afford to spare large IPv4 address spaces for operating a darkspace. We then will probably see more distributed forms of darkspaces that consist of combinations of smaller darkspaces or combine darkspace with lightspace networks, which contains active hosts. We can distinguish the following structures:

- Distributed Darkspaces: Instead of operating one large darkspace multiple smaller dark address blocks are combined.
- Greyspaces: Most networks have some dark spots in form of unused IP addresses or unused ports. Monitoring traffic to those addresses and ports in otherwise used subnets is useful to detect unusual activities.
- Dynamic Darkspaces: Darkspace in which IP addresses are temporarily used. Addresses are temporarily light up by connecting an active host.

The combination of data from multiple distributed darkspaces is useful to investigate the global distribution, scope and development of large events. Using distributed small darkspace also provides a higher address diversity. Combining darkspace data with other sources, e.g., from lightspace, honeynets or intrusion detection systems helps to get further insight into incidents.

The Internet Motion Sensor Architecture ([8], [4], [9]) is a distributed darkspace that consists of darkspace monitors at different networks. The architecture contains some lightweight responders in order to extract more information from malware sources.

The authors of [12] investigate the spatial correlation between two adjacent address blocks. Their results show that TCP SYN traffic time series shows spatial correlation in of packets arrivals between two neighboring address blocks.

But combining small fractions of distributed dark address spaces introduces new challenges for inferring size and scope of attacks. Furthermore, some synchronization is required to correctly correlate events from different locations. The clocks of the involved measurement devices should be synchronized. Analysis intervals, classification rules and analysis techniques should be the same.

If darkspace data is shared among different organizations it may be needed to remove sensitive information first. For instance backscatter traffic contains the IP addresses of vulnerable/attacked hosts and therefore contains critical information that should not be revealed to others.

Greyspace analysis [14] may become more popular in future. Most networks have some dark spots in form of unused IP addresses that can be utilized for darkspace analysis purposes. The presence of active hosts in the same local network will probably direct more traffic of interest to greyspaces than to a completely dark network

of the same size. A nice technique is also to introduce a few light responders to the network that act as honeypots to gather further details about onging attacks.

In dynamic darkspaces, addresses become temporarily active. The authors of [17] report results from a dynamic darkspace. They show that after lightening up one host, the vast majority of the traffic was directed to that host. This behavior continued even after they darkened the address again.

4.1 Darkspace analysis in IPv6 networks

A main feature of IPv6 is its much larger address space compared to IPv4. An IPv6 address consist of 128 bit. In most cases the last 64 bit of an IPv6 address are used to address a host (or interface) within a network [15]. The huge IPv6 address space gives us plenty of networks and addresses within a network to be used as dark spaces. But due to the huge address space of 2^{64} potential addresses (instead of 2^8 addresses in an IPv4 subnet), random scanning of an IPv6 network becomes impractical. Assuming that scanning 2^8 addresses takes 1 minute, scanning 2^{64} addresses takes 2^{56} times longer which leads to approximately 140 billion years.

Nevertheless, this IPv6 advantage only applies if addresses within a subnet are really assigned randomly and attackers cannot gain information about active addresses from other sources. Any structured numbering of hosts makes it easier for an attacker to guess or deduce active IPv6 addresses. The use of stateless autoconfiguration based on MAC addresses makes it even worse. The address then include the vendor IDs which that can be easy to guess.

New IPv6 scanning methods are discussed in [7]. Due to the different scanning strategies for IPv6, purely random scanning is much less likely to occur in IPv6 networks. We expect to see not much activities from process that use random addresses in IPv6 darkspaces. This reduces the usability of darkspace traffic for attack detection and malware analysis immense. IPv6 darkspaces have been investigated in [11] and [16]. Both show only very few activities from malicious activities in IPv6.

Nevertheless, it can make sense to use specifically crafted darkspace structures in IPv6. One option is to leave a few addresses dark when assigning IP addresses in a continuous scheme. This would provide information about scanning activities in the network after attackers gained insight into the addressing scheme.

References

1. Emile Aben. Conficker/Conflicker/Downadup as seen from the UCSD Network Telescope. Technical report, CAIDA, February 2009.
2. Ejaz Ahmed, Andrew Clark, and George Mohay. Characterising Anomalous Events Using Change - Point Correlation on Unsolicited Network Traffic. In Audun Jøsang, Torleiv Maseng, and Svein Johan Knapskog, editors, *Identity and Privacy in the Internet Age*, volume 5838, pages 104–119. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

3. Ejaz Ahmed, Andrew Clark, and George Mohay. Effective Change Detection in Large Repositories of Unsolicited Traffic. In *Fourth International Conference on Internet Monitoring and Protection, 2009. ICIMP '09*, pages 1–6. IEEE, May 2009.
4. Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. The Internet Motion Sensor: A Distributed Blackhole Monitoring System. In *Proceedings of Network and Distributed System Security Symposium (NDSS 05)*, pages 167—179, 2005.
5. Michale Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, and Sushant Sinha. Practical Darknet Measurement. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 1496–1501. IEEE, March 2006.
6. Nevil Brownlee. One-way Traffic Monitoring with iatmon. In *13th Passive and Active Measurement Conference (PAM 2012)*, 2012.
7. Tim Chown. IPv6 Implications for Network Scanning. RFC 5157 (Informational), March 2008.
8. Evan Cooke, Michael Bailey, Z. Morley Mao, David Watson, Farnam Jahanian, and Danny McPherson. Toward understanding distributed blackhole placement. In *Proceedings of the 2004 ACM workshop on Rapid malcode*, WORM '04, page 54–64, Washington DC, USA, 2004. ACM. ACM ID: 1029627.
9. Evan Cooke, Michael Bailey, David Watson, Farnam Jahanian, and Jose Nazario. The Internet Motion Sensor: A Distributed Global Scoped Internet Threat Monitoring System. Technical report, Electrical Engineering and Computer Science Department, University of Michigan, 2004.
10. Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C. Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of Country-wide Internet Outages Caused by Censorship. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurements (IMC)*, IMC '11, pages 1–18, New York, NY, USA, 2011. ACM.
11. Matthew Ford, Jonathan Stevens, and John Ronan. Initial Results from an IPv6 Darknet. *Internet Surveillance and Protection, International Conference on*, 0:13, 2006.
12. Kensuke Fukuda, Toshio Hirotsu, Osamu Akashi, and Toshiharu Sugawara. Correlation Among Piecewise Unwanted Traffic Time Series. In *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, pages 1–5. IEEE, December 2008.
13. Uli Harder, Matt W Johnson, Jeremy T Bradley, and William J Knottenbelt. Observing Internet Worm and Virus Attacks with a Small Network Telescope. *Electron. Notes Theor. Comput. Sci.*, 151(3):47–59, June 2006. ACM ID: 1706686.
14. Warren Harrop and Grenville Armitage. Defining and Evaluating Greynets (Sparse Darknets). In *The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary*, pages 344–350. IEEE, November 2005.
15. Robert M. Hinden and Stephen E. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006. Updated by RFCs 5952, 6052.
16. Geoff Huston. IPv6 Background Radiation. In *North American Network Operators Group Meeting (NANOG 50)*, October 3-6 2010.
17. Jeff Janies and Michael Patrick Collins. Darkspace Construction and Maintenance. In *CERT FloCon 2011 Proceedings*, Salt Lake City, UT, January 2011. CERT.
18. David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. Inferring Internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, May 2006. ACM ID: 1132027.
19. David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity'. In *Proceedings of the 2001 USENIX Security Symposium, Washington D.C.*, pages 9–22, August 2001.
20. Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of Internet Background Radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, page 27–40, Taormina, Sicily, Italy, 2004. ACM. ACM ID: 1028794.
21. Songjie Wei and Jelena Mirkovic. Correcting Congestion-based Error in Network Telescope's Observations of Worm Dynamics. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet measurement (IMC 2008)*, IMC '08, pages 125–130, New York, NY, USA, 2008. ACM.
22. Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. Internet background radiation revisited. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, page 62–74, Melbourne, Australia, 2010. ACM. ACM ID: 1879149.
23. Tanja Zseby. Entropy in IP Darkspace Data. In *CERT FloCon 2012 Proceedings*, Austin, TX, January 2012. CERT.

Flow Data Collection in Large Scale Networks

Pavel Čeleda[1] and Vojtěch Krmíček[2]

[1] Masaryk University, Institute of Computer Science
Botanická 68a, 602 00 Brno, Czech Republic
celeda@ics.muni.cz
[2] CESNET, z.s.p.o.,
Zikova 4, 160 00 Prague, Czech Republic,
krmicek@cesnet.cz

Abstract. In this chapter, we present flow-based network traffic monitoring of large scale networks. Continuous Internet traffic increase requires a deployment of advanced monitoring techniques to provide near real-time and long-term network visibility. Collected flow data can be further used for network behavioral analysis to indicate legitimate and malicious traffic, proving cyber threats, etc. An early warning system should integrate flow-based monitoring to ensure network situational awareness.

1 Introduction

Detailed traffic statistics are necessary to provide a permanent network situational awareness. Such statistics can be complete packet traces, flow statistics or volume statistics. A trade-off must be chosen between computational feasibility and provided level of information to efficiently handle high-speed traffic in large scale networks.

- *Full packet traces* traditionally used by traffic analyzers provide most detailed information. On the other hand the scalability and processing feasibility for permanent traffic observation and storing in high-speed networks is an issue including high operational costs.
- *Flow statistics* provide information from Internet Protocol (IP) headers. They do not include any payload information, however we still know from IP point of view who communicates with whom, which time, etc. Such approach can reduce up to 1000 times the amount of data necessary to process and store. Using flow we are able even to monitor encrypted traffic.
- *Volume statistics* are often easy to obtain in form of Simple Network Management Protocol (SNMP) data. They provide less detailed network view in comparison with flow statistics or full packet traces and do not allow advanced traffic analysis.

We use flow data for their scalability and ability to provide a sufficient amount of information. Flow-based monitoring allows us to permanently observe both small end-user networks and large National Research and Education Network (NREN) backbone links.

2 Flow-Based Monitoring

Measurement of IP flow statistics was first introduced by Cisco Systems [2] in 1996. *NetFlow protocol* introduced de facto standard for today's flow monitoring. NetFlow is used to export flow information from so called *exporter* to *collector*. The flow is defined as a sequence of consecutive packets with the same IP addresses, ports and protocol number.

NetFlow is traditionally used for routing optimization, application troubleshooting, traffic mix monitoring, accounting and billing, and others. Besides these running-up applications new utilization attracts the attention including detection of security incidents and Denial of Service (DoS) attacks, already embedded in some collectors. Network Behavior Analysis (NBA) is an alternative flow-based approach to traditional pattern matching to detect cyber threats, e.g. Advanced Persistent Threats (APT).

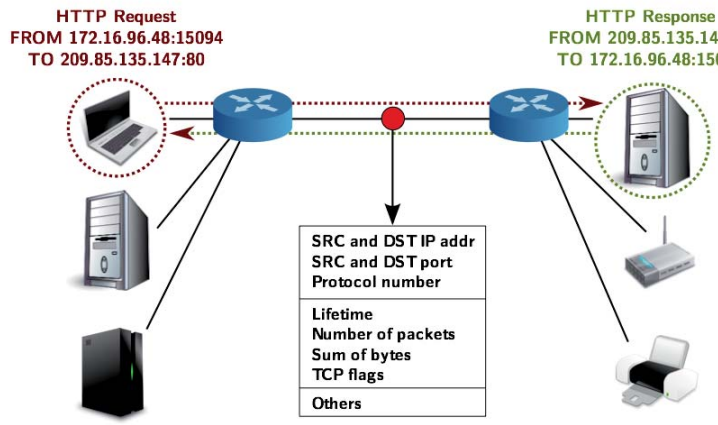
2.1 Definition of IP Flow

In general, flows are a set of packets which share a common property. The most important such properties are the flow's endpoints. The simplest type of flow is a 5-tuple, with all its packets having the same *source IP address*, *destination IP address*, *port numbers*, and *protocol* (see Figure 1). Flows are unidirectional and all their packets travel in the same direction. The flow begins when its first packet is observed. The flow ends when no new traffic for existing flow is observed (inactive timeout) or connection terminates (e.g. TCP connection is closed). An active timeout is time period after which data about an ongoing flow are exported. Statistics on IP traffic flows provide information about *who* communicates with *whom*, *when*, *how long*, using *what protocol* and *service* and also *how much data* was transferred.

3 Flow Exporters

NetFlow can be enabled on routers which constitute primary source of NetFlow data today. On the other hand utilization of standalone dedicated systems such as dedicated probes seems to have several benefits. Offloading of resource intensive flow measurement to dedicated probe is probably the most important one. When NetFlow is enabled the routers often suffer of huge system load. Dedicated probe allows routers to perform their primary task, i.e. to route packets and keep mission critical applications up and running. A wide variety of network devices are capable of generating flow. We mention three basic categories of flow exporters:

– *Routers, switches, and firewalls* observe all traffic going through the network. They are in a unique position to generate detailed flow data. However, in some environments (e.g. on high-speed backbones) they are not able to process all




```
Flow start   Duration Proto   Src IP Addr:Port         Dst IP Addr:Port   Flags  Packets  Bytes
09:41:21.763   0.101  TCP     172.16.96.48:15094 -> 209.85.135.147:80     AP.SF    4       715
09:41:21.893   0.031  TCP   209.85.135.147:80     ->   172.16.96.48:15094  AP.SF    4      1594
```

Fig. 1. The IP flow statistics describing endpoints communication.

packets. So *sampled NetFlow* was introduced by Cisco. When sampling is used only one packet out of n is processed. The flow data are less accurate and a further flow processing can be affected e.g. sampling effect on anomaly detection methods.

Many vendors provide an equivalent to Cisco NetFlow technology on their devices, however some of them use a different name for it, e.g. jflow, clowd, NetStream. The flow support is implemented in software (lower performance, sampling required) or in dedicated hardware (high performance, some flow elements may be missing e.g. TCP flags).

– *Dedicated flow probes* were developed to overcome limitations (costs, availability, and functionality) of router based flow exporters. The most well known exporters include *nProbe*, *yaf*, and *FlowMon*. They are typically implemented as open-source software (Linux, BSD) using commodity hardware (server, network interface cards). The probes use kernel TCP stack bypass to process more traffic in comparison to standard Packet Capture (PCAP) interface. Further the Receive-Side Scaling (RSS) is used to distribute network traffic to multiple cores. To achieve line-rate processing the hardware acceleration is used. There are special Field-Programmable Gate Array (FPGA) based cards.

  Test Access Port (TAP) devices or Switched Port Analyzer (SPAN) ports must be used to provide input for probes. TAP devices are non-obtrusive and are not detectable on the network. They send a copy (1:1) of all network packets to a probe. In case of failure the TAP has built-in fail-over mode. The observed line will not be interrupted and will stay operational independent on any potential probe failure.

In case of SPAN, we must enable the port mirroring functionality on a router/switch side to forward network traffic to monitoring device. However, we must take in count some SPAN port limits. Detailed comparison between using TAP devices or SPAN ports is described in [9].
- *Virtual flow probes* are special case of dedicated probes. They are used in virtualized environments, to monitor the traffic passing through a virtual switch or a virtual TAP. They are available as a virtual appliance.

3.1 Flow Export Formats

Several *NetFlow* versions were introduced by Cisco. The most common are version 5 and version 9. NetFlow version 5 uses fix data format and is restricted to IPv4 flows. NetFlow version 9 [4] introduced templates to describe flow data. Version 9 supports IPv6, MPLS, VLANS, and MAC addresses.

NetFlow version 9 was the basis for Internet Protocol Flow Information eXport (IPFIX) [5]. *IPFIX* is developed and promoted by Internet Engineering Task Force (IETF). It provides a standard for IP flow information from routers, probes, and other devices. There is still (as of 2012) low IPFIX support, especially advanced IPFIX functions are missing. Most implementations support only NetFlow version 9 subset in IPFIX protocol.

3.2 Flow Application Extensions

Flow data provide information from *data link layer* (L2), *network layer* (L3), and *transport layer* (L4) of Open Systems Interconnection (OSI) model. It is no longer possible to rely on port numbers to identify applications. Typically HTTP traffic (TCP port 80) can pass through most firewalls and presents a way how to tunnel data. HTTP became the new Transmission Control Protocol (TCP) and a traffic passing over HTTP must be inspected.

The *application visibility* is crucial to prevent all kind of unwanted traffic. IPFIX provides Enterprise Information Elements to store *application layer* (L7) information. AppFlow [3] is an example how to describe the actual applications in use within the flow. Similar functionality provides Cisco NBAR with Flexible NetFlow, Palo Alto firewalls, and other application-aware flow exporters.

4 Flow Collectors

IP flow generation represents first important step to be able to monitor a large scale network. Once we are able to export flow data from the particular metering points in the network, next step is to collect it, store it and provide suitable tools for further flow data analysis. In this part, we present current approaches to storing IP flow data and discuss frequently used operations needed during data analysis.

4.1 Flow Collectors in General

The main role of collector is to store flow data for prospective further analysis. Beside this, there is a couple of other features and roles, which should be implemented at the collector side to provide full support for network administrator work, including further manipulation with the data like flow listing, flow filtering, flow aggregation and also support for automatic flow data analysis.

There is a large number of existing collectors available to install and deploy. We can choose either commercial solution with full technical support or decide to use open-source project with technical support usually provided by collector community. There are a number of open-source collectors with wide spectrum of features, large user community and active development. We suggest choosing some of these open-source projects. In the following, we will illustrate collector features on the NetFlow Sensor (NfSen) example [7], used in our backbone and campus network.

The main challenge collectors are facing to, is a storing and processing of a large amount of IP flows. With the increasing speeds of modern computer networks, this amount is growing up and the requirements for data storage are demanding. Five minutes of IP flow data from the backbone 10 Gbps link with halfway load represents e.g., five billions of flows to store. Therefore we are not able to store full IP flow data in long term history and we need to replace the old data with a new one after, e.g., a couple of months. There are several ways how to reduce the amount of flows, but all these approaches (sampling, data aggregation) led to losing some amount of information contained in full flow data.

To be able to cope with this huge amount of flow data, the collector has to have an effective data storage back-end. The manipulation with the stored flow data should provide fast methods how to search/filter flow data and these tasks become nontrivial with the huge amounts of flows. We can see basically two types of data storage for flow data:

- *Relation database (SQL) approach* – advantages of this approach are well-known database mechanisms with full support for *querying*, *searching*, and *indexing*. Contrary to this support, this type of databases was not designed to work with such huge amounts of data. If we use this solution in backbone network, we will face non-trivial problems with the size of database, huge times needed for database *reorganization*, *querying*, *making indexes*, *integrity*, etc. Therefore this type of flow data storing is suitable for the smaller networks.
- *Flat file approach* – in this case, the IP flow data are stored in files and dumped directly to the disk. This approach is suitable for storing large amounts of flow data. It does not need any further maintenance and does not consume too much processing power. On the other hand, we need to access such data sequentially and there are usually no indexes and metadata over such files. However in the case of flow collection from large networks, this approach represents more effective approach compared to SQL approach (see [8]).

Supported format of exported flow data differ in various collectors. We can see support of NetFlow version 5 and also NetFlow version 9 in majority of collectors. However, there are sometime problems with full support of template mechanism used by NetFlow v9. As discussed in Section 3.1 on page 33, the IPFIX format will substitute current NetFlow v5/v9 formats, but its support in currently used flow collectors is problematic due to the large possibilities of IPFIX format extendability.

4.2 Processing Data with Flow Collector

In the following, we describe the most frequent operations performed with flow data at the collector side by network administrator. We identify a number of standard tasks performed with flow data and illustrate them on the example of NfSen collector.

Data Storage and Redistribution The first step performed by each collector is to acquire flow data from the network and consequent storing to the disk. Depending on the type of data storage system, there could be performed further data reorganization, redistribution, etc. In the case of NfSen collector, we can define a various parameters, e.g., where to store data, how long should be time window for storing data and where to replicate an identical copy of flow data acquired from network. Also we can define, what extensions to NetFlow v9 we want to store, e.g., VLAN IDs, AS numbers, MAC addresses, etc.

Command Line Interface Once we have stored flow data from the network at the collector side, we need to perform its analysis. For this purpose, collectors have command line interface (CLI), web front-end or both, providing the access to the flow data. Typically, we need to perform one or more of the following actions:

- *List flows* – we need to see flows itself, with all information stored inside them. A possibility to define output format of flows – which fields from flow we want to list – would help in the flow analysis. NfSen collector supports CLI flow analysis with fully configurable output format.
- *Filter flows* – filtering is used very often to find out particular IP addresses or communication in the network. For this case, the collectors have defined filtering syntax, which should provide a possibility to filter out flows satisfying various conditions, or in the case of SQL collectors, we use SQL queries. NfSen collector provides syntax similar to Berkeley Packet Filter (BPF) with various filtering possibilities. Also the time needed to obtain corresponding flows is crucial for efficient work with collector.
- *Flow aggregation* – possibility to aggregate flows by various fields is used often to obtain overview of the network traffic in monitoring network (e.g., aggregation by ports, subnets/IP addresses, protocols, etc.).

– *Top talkers* – this function provide a quick overview of the most active hosts in network and can be used for revealing possible attackers/victims.

Web Interface Although CLI provides detailed access to the flow data, it does not contain any graphical representation of the data. Therefore, collectors usually dispose with *graphical front-end*, representing network traffic in the form of graphs. Network administrator is able to see any outages or discrepancies in the observed network easily in the graphs, compared to the CLI.

In case of NfSen collector, there is a variety of graphs representing flow data from different views (by protocols and for different time periods) and also it provides direct access through the web interface to the flow data. Therefore, once network administrator identifies a network problem in the graph, she can directly list flows corresponding to this issue and perform further analysis.

Beside the standard set of graphs, NfSen collector provides also a possibility to define profiles. *Profiles* are defined by the flow data sources and filters applied to them. As a result, corresponding data matching the filter are stored separately and new graphs are generated. Using this feature, we are able to define profiles for e.g., various services or types of traffic (HTTP, DNS, routing data, etc.) in the network and we can inspect it separately.

Automatic Flow Data Processing Beside the basic flow data storing and listing/displaying/filtering, the collectors provide tools for automatic flow data processing. Therefore, network administrator doesn't have to check manually all the data, but he can define various conditions or use *automatic analysis* methods to inspect current flow data and in the case of some attack or discrepancy, an alarm can be triggered automatically.

NfSen collector provides *alerting* tool, with the possibility to define various conditions and also consequent actions (sending email, starting particular plugin). Therefore, the network operator can be noticed automatically about, e.g. network outages.

Extension Possibilities Although flow collectors have a lot of functionality integrated inside, we need to perform some customized post processing of flow data often. For this reason, the extension interface providing standardized access to the flow data can be used with advantage. NfSen collector provides well defined interface for adding new *plugins*, which are able to analyse flow data stored by collector, perform further processing and also display results through the standard collector web interface.

5 Monitoring Use Cases

In this part, we describe two large scale networks *(i)* campus network of Masaryk University and *(ii)* the backbone network of CESNET [1]. We show differences between these two networks, consisting mainly in the type of a monitored traffic. In the case of campus network, we monitor the traffic ending or originating inside the network. In the case of backbone network, we are monitoring transit traffic going through the network. Due to these differences, we would also obtain different flow data and we should deploy different configurations of flow monitoring system.

5.1 Campus Network

In this scenario, we show campus network containing about 15 000 networked hosts. The Internet connection is realized through two 10 Gbps links. One part of traffic does not leave the campus network itself, because it is targeted to the other hosts inside the network. Other part is incoming and outgoing traffic routed to CESNET (public Internet).

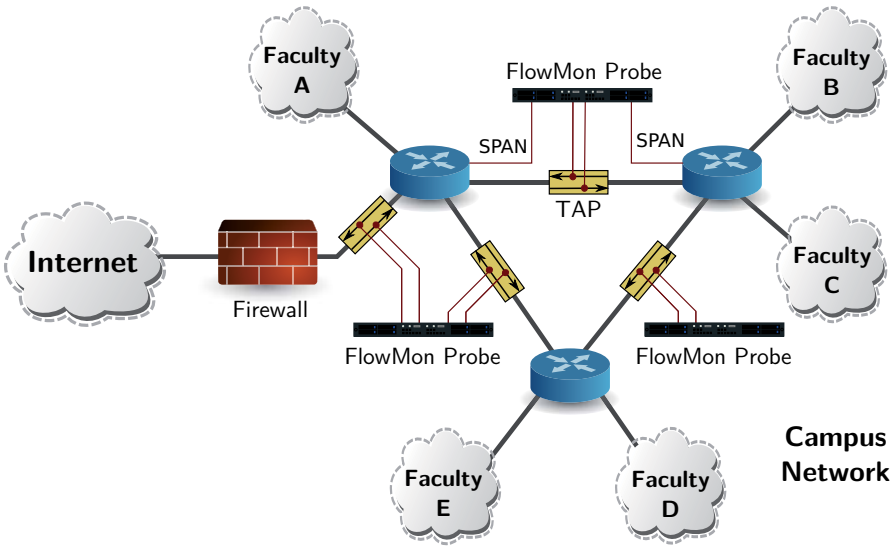


Fig. 2. Flow probes deployment in the campus network.

Figure 2 illustrates possible deployment of flow probes in the campus network. One probe is deployed to monitor traffic at the border of the campus network. Analysing this type of traffic, we are able to disclose e.g., security incidents and attacks against campus network and also malicious traffic and attacks going from

campus network to the Internet. We used flow analysis to *reveal previously unknown Chuck Norris botnet* [6] attacking poorly configured embedded devices (see Figure 3).

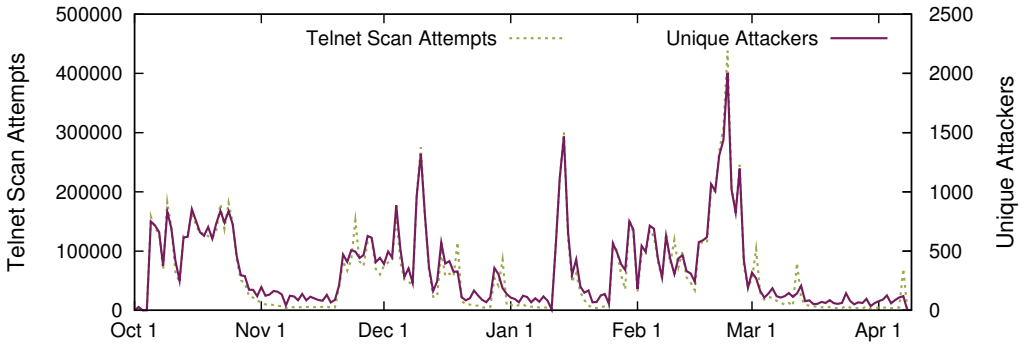


Fig. 3. Unique attackers and attacks on TCP port 23 in the campus network.

Further the probes are deployed between particular campus faculties, monitoring the traffic generated inside the network and transferred among the campus hosts. Analysing this type of traffic, we are able to observe amount of traffic transferred between particular parts of campus, the load of network links in campus networks and also security incidents inside the campus network.

The list of situations monitored by network administrators includes: security incidents, state of particular network services, network load at the particular parts of campus network, user statistics and accounting, etc.

5.2 Backbone Network

The second scenario demonstrates the large backbone network of CESNET interconnecting academic and research institutions in Czech Republic. The majority of network traffic is *transit traffic*, going through the backbone network to other destination and not ending here.

Figure 4 illustrates a deployment of dedicated flow probes (FlowMon) in this type of network. We deploy flow probes connected via TAP devices at all 10 Gbps links providing connection to foreign countries and one probe inside the backbone network. Therefore we are able to inspect both the traffic incoming and outgoing from the foreign networks and also traffic originating in the domestic networks.

Analysis of backbone data is focused on different aspects. In this case, we are interested in the amounts of the network traffic transferred between particular networks and to/from foreign countries. Also the monitoring of link outages and rout-

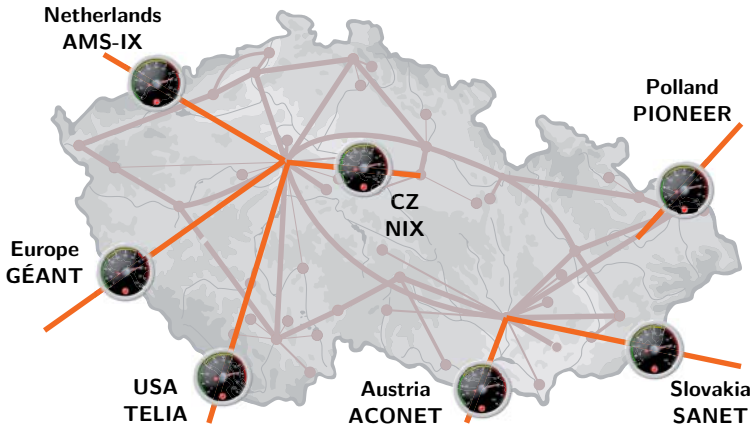


Fig. 4. Map of the Czech Republic with the FlowMon probes deployment in the CESNET's backbone network. Probes are monitoring international 10 Gbps links.

ing traffic could help network administrators. The monitoring of large scale security incidents is important too.

Figure 5 represents one week traffic observed at SANET link to Slovakia. We monitor both traffic directions (from abroad to backbone network and vice versa). There are remarkable *diurnal* and *weekly patterns* - lower traffic on weekend and the maximum load of network during Wednesday and Thursday.

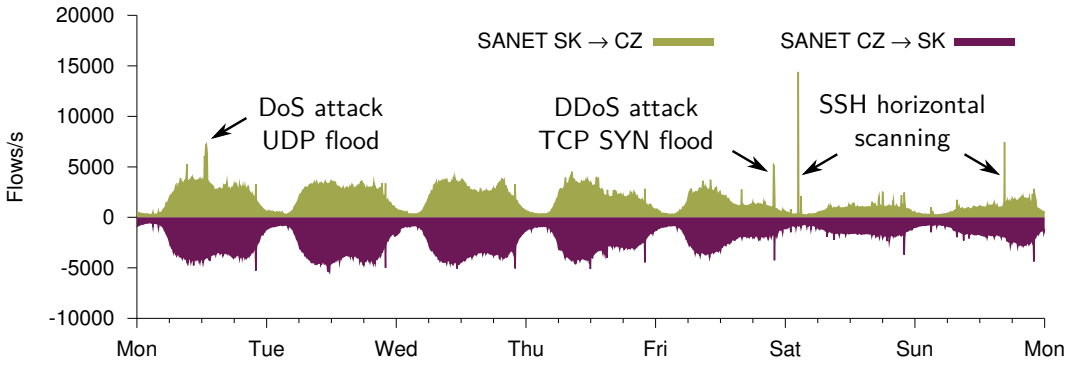


Fig. 5. Diurnal traffic pattern and four security incidents in the SANET 10 Gbps link interconnecting Czech Republic and Slovakia – time window April 23-30, 2012.

Besides the diurnal and weekly patterns, we can observe four massive *security incidents* at the Figure 5. First one represents massive DoS attack (UDP flood consisting of 6.3 M flows) launched from Slovakia against a university in the Czech Republic. Next attack is a large distributed DoS (DDoS) attack against public web

hosting provider consisting of 2.9 M attacker flows. Two another serious security issues follow – massive SSH scans consisting of 4.3 M and 1.9 M flows originating at Slovakia high school and targeting more than 5 M possible victims worldwide. Although these massive incidents are easily detectable, majority of network attacks and malicious activities are not so intensive and therefore we need to perform advanced analysis of flow data to be able to reveal them.

6 Summary

This chapter gave an overview about flow monitoring in large scale networks. We described flow exporters, data export formats, and flow collectors. Two use cases showed today's deployment of flow-based monitoring in campus and backbone network. We also showed flow data of botnet malicious activity and various attacks at international backbone link.

References

1. CESNET: *Czech National Research and Education Network operator*, `http://www.ces.net/`, 2012.
2. Cisco: Cisco IOS NetFlow, `http://www.cisco.com/go/netflow`, 2012.
3. Citrix: *AppFlow Specification*, `http://www.appflow.org/`, 2012.
4. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), IETF, 2004. `http://www.ietf.org/rfc/rfc3954.txt`
5. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard), IETF (2008). `http://www.ietf.org/rfc/rfc5101.txt`
6. Čeleda, P., Krejčí, R., Vykopal, J., Drašar, M.: Embedded Malware – An Analysis of the Chuck Norris Botnet. In: Proceedings of the 2010 European Conference on Computer Network Defense, Berlin, 2010.
7. Haag, P.: *NfSen – NetFlow Sensor*, `http://nfsen.sourceforge.net/`, 2012.
8. Hofstede, R., Sperotto, A., Fioreze, T., Pras, A.: *The Network Data Handling War: MySQL vs. NfDump*. 16th EUNICE/IFIP WG 6 6 Workshop (EUNICE 2010), Trondheim, 2010.
9. Zhang, J., Moore, A.: *Traffic trace artifacts due to monitoring via port mirroring*. In Proceedings of the Fifth IEEE/IFIP E2EMON, pages 1-8, 2007.

Flow-based Brute-force Attack Detection

Martin Drašar[1], Jan Vykopal[1], and Philipp Winter[2]

[1] Masaryk University, Institute of Computer Science
Botanická 68a, 602 00 Brno, Czech Republic
{drasar,vykopal}@ics.muni.cz
[2] Karlstad University, Department of Computer Science
Universitetsgatan 2, 651 88 Karlstad, Sweden
philipp.winter@kau.se

Abstract. Brute-force attacks are a prevalent phenomenon that is getting harder to successfully detect on a network level due to increasing volume and encryption of network traffic and growing ubiquity of high-speed networks. Although the research in this field advanced considerably, there still remain classes of attacks that are hard to detect. In this chapter, we present several methods for the detection of brute-force attacks based on the analysis of network flows. We discuss their strengths and shortcomings as well as shortcomings of flow-based methods in general. We also demonstrate the fragility of some methods by introducing detection evasion techniques.

1 Introduction

In recent years, network security research started focusing on *flow-based* attack detection in addition to the well-established *payload-based* detection approach. Instead of only looking for malicious activity in the actual packet data, network flows are also considered for analysis [8]. This is not surprising since the amount of data one has to fight with is drastically reduced and the attacks visible in flow data tend to complement the attacks that we strive to find in network payload.

In this chapter, we give a compact overview of current research in this field with respect to brute-force attacks. We propose five detection techniques and shed light on the shortcomings inherent to the flow-based attack detection approach.

This chapter is divided into five sections. The rest of this section highlights the difference in attack orchestration and discusses detection in encrypted traffic. Section 2 describes five different approaches to flow-based intrusion detection that reveal brute-force attacks. Limitations imposed by the nature of flows are summarized in Section 3. Four detection evasion techniques are then outlined in Section 4. The chapter is recapitulated in Section 5. Flow data collection in large scale networks is extensively covered by Chapter *Flow Data Collection in Large Scale Networks* on page 30.

1.1 Noisy Versus Stealthy Attacks

Attacks that occur in a network can be roughly divided into two categories, depending on their impact on traffic patterns. On the one side there are *noisy attacks* that

disrupt these patterns significantly. One example is port scans that often precede actual attacks [9]. Such attacks are very easy to detect since all that is needed is to look for a sudden increase in traffic volume. Noisy attacks are useful to penetrate networks that are not sufficiently protected and to estimate defense capabilities of particular networks. They can also be used as a cover for stealthy attacks running simultaneously. Any exposed network is likely to be target sooner or later, so it is easy to gather real life examples.

On the other side, there are *stealthy attacks*. These attacks are much harder to gather and examine as they by virtue try to remain undetected. Stealthy attacks have to be crafted for a target network and must reflect its detection capabilities. Staying under the radar also means that the attack is generally slower and that it has to run longer.

1.2 Detection of Attacks in Encrypted Traffic

Various secured protocols, services and applications became more and more popular in recent years. Besides services such as SSH, even web applications provided by Google or Facebook are currently accessible over HTTPS. Furthermore, user authentication via secured communication channels is becoming a standard these days.

With the rise of encrypted traffic, the traditional approach to network-based intrusion detection is becoming *ineffective*. Packet payload which is searched for signatures of known attacks by deep packet inspection is opaque, only packet headers can be analyzed. Therefore, flow-based detection is one of the possible ways to deal with encrypted traffic.

2 Detection of Brute-force Attacks

Brute-force attacks are most frequently detected at the host level by inspecting access logs. If the predefined number of unsuccessful login attempts is reached, an alert is fired, the attacker blocked or other attempts significantly delayed. This approach is effective, even for distributed attacks. The main drawback is that it does not scale well.

We present five detection approaches that profit from the scalability of network flows. The first is a simple analogy of pattern matching known from deep packet inspection. The second approach extends the first one by searching for *similar* traffic instead of fixed patterns. The following two exploit periodicity and the even distribution of attacks in time. And the last one finds abrupt changes in entropy time series.

2.1 Signature-based Approach

Similarly to pattern matching in deep packet inspection, signatures can be used in flow-based intrusion detection too. The flow-based signatures describe network traffic by specific values, or ranges of values, of flow features and computed statistics. The signatures are then searched for in acquired flows. This is done in separate time windows, typically when exported flows are sent from the collecting process to the metering process. So this simple approach does not consider changes of the monitored traffic in time.

Concerning brute-force attacks, the relevant signature can be comprised of features and statistics describing both requests and replies thanks to the interactive nature of the attacks. The requests carry attempted credentials and the replies information about whether the login was successful or not.

Method First, the most popular attacked services such as SSH, Telnet, RDP or web applications using HTTP or HTTPS are run on mostly well-known network ports such as TCP/22, TCP/23, TCP/3389, TCP/80 or TCP/443. Second, the source port of the client (attacker) request, i. e. the destination port of the reply, is usually greater than 1024. Third, login attempts and server replies have a specific (range of) size and duration. These characteristics can be captured by the number of packets and bytes of a flow, its duration or statistics: packets per second, bytes per second and bytes per packet. To sum it up, the signature of an attacker's attempt of SSH authentication may be defined as follows: protocol = TCP, source_port $>$ 1024, destination_port = 22, packets $>$ 10, packets $<$ 30, bytes $>$ 1400, bytes $<$ 5000, duration $<$ 5 s.

Next, selected features of flows matching the given signatures may be analyzed again. For example, in order to determine the number of unique attackers and victims (source and destination IP addresses).

Finally, the number of matching flows is counted for each attacking IP address and if the predefined threshold is reached, an alert is fired. The threshold should express an anomalous number of login attempts in the time window (e. g. 10 in a 5-minute time window for the sample SSH signature above).

Discussion The signatures can be implemented as a chain of filters for the nfdump tool [3] or as a decision tree [10].

In 2010, we have deployed a simple signature for SSH attacks in the campus network of Masaryk University to find suspicious hosts conducting SSH attacks. The network consists of about 15 000 networked hosts with public IP addresses including hundreds of SSH servers. The network is open and naturally attracts attackers' attention. The signature itself matched traffic of a few thousand of attackers, but also a few tens of possibly benign hosts from our network. These false positives were caused mainly by hosts connected to a grid and Nagios servers. To eliminate these

false positives, we employed the fact that the majority of attack flows is produced by attackers aiming at more hosts within one attack and that the attacks are preceded by scanning the port TCP/22 [9].

In conclusion, the signature-based approach is very straightforward, simple and effective, but for operational use (to eliminate false positives) it is necessary to employ other data sources supporting or contradicting the result.

2.2 Similarity-based Approach

Deriving signatures as described above is a time-consuming process. Existing signatures need maintenance as tools and systems generating monitored traffic are evolving and traffic patterns are changing. Additionally, "0-day" attacks are not recognized. We try to address these issues by searching for *similar* flows instead of matching specific signatures. We believe that the similarity of traffic can point to machine-generated traffic, for instance brute-force attacks.

Method First, all incoming flows are clustered in a separate time window to isolated groups of similar flows. The similarity is measured by the distance of particular flows (points) in space defined by flow features and statistics. We need to choose a *distance metric function*, its input parameters, i. e. suitable flow features and *radius* used for determination if the flow (point) belongs to groups of flows (points) that are close to each other. For example, we can define points p_{id} representing flows in two-dimensional space as follows: $p_{id} = (pkt, byt)$, where pkt is the number of packets and byt the number of bytes in the flow, and id is a flow identification used in further processing. The distance metric function may be, e. g., the Euclidean metric d given by the Pythagorean formula: $d(p_1, p_2) = \sqrt{(pkt_2 - pkt_1)^2 + (byt_2 - byt_1)^2}$ and the radius a float number.

Second, we assume that flows representing malicious traffic are grouped in clusters[3] with a large member (point) count and flows representing benign traffic form clusters with a low member count, because this traffic consists of flows with very variable properties. These clusters can also be excluded from further processing.

Third, IP addresses of flows (points) within each cluster are inspected and the type of the attack is determined. If the cluster contains randomly distributed IP addresses, it may indicate benign traffic. On the contrary, if it is possible to find the same source or destination addresses, it may point out to a multiple or a distributed attack.

Finally, the IP address is classified as the attacker if it generated more than the predefined number of flows.

Discussion This is a more generic approach and its detection capability essentially depends on a chosen clustering algorithm and its parameters.

[3] Some clusters may contain traffic generated by port scanning and some other actual password guessing.

2.3 Detection of Automated Actions

Most of the brute-force attacks that we have observed in practice exhibit one similarity that we attribute to their automated behavior: the intensity of an attack from one source *remains relatively constant and periodic* during its course. This attribution is supported by our knowledge of available brute-forcing tools that generally allow their users to set the attack intensity only by specifying the number of attack tasks running in parallel.

Traffic with such property is naturally not unique to brute-force attacks — querying the NTP server, various protocols' keep-alives, IM, etc. behave in a similar way — however, this communication is usually directed to well-known machines or ports that are generally not targets of attacks. These can be thus easily discarded beforehand.

Time Window Heuristic Detection of traffic with almost constant intensity can be done using a simple heuristic. Any machine attacking with constant intensity and with zero or fixed delays between attack attempts will create only slightly varying number of flows in any two time windows. This number can be influenced by e.g. network conditions or machine slowdowns.

Figure 1 illustrates two attacks that are both periodic and constant-intensive. There are on average 250 attempts every 6 minutes. The burst attack does not have fixed delays between attacks as it concentrates these attempts into the first 2.5 minutes, whereas the continuous attack distributes the attempts evenly.

The ability to identify this traffic as potentially malicious using the aforementioned heuristic largely depends on two properties of the used time windows. First, there is a size of a time window. As can be clearly seen in Figure 1, short time windows (dark rectangles) may be good enough to detect an attack that does not vary much, but may fail with more complicated patterns. Longer time windows (light rectangles) may detect even the burst attack, but may fall short for attacks that do not last long. Second, there is the relative temporal distance of time windows. Fixed distances can suffer from local non-uniformity of attack intensity when a time window is shorter than the execution of all attack tasks running in parallel. Random distances with potential overlapping seem like a better choice to counter this scenario.

Fourier Transform Another approach to identify almost constant periodic traffic is to employ methods of signal processing. The basic premise is that a network traffic trace can be viewed as a signal that can be processed further. This approach was already used in [2], [5] and by others to detect network anomalies. These authors basically used signal processing methods to find deviations in traffic to identify an attack. In this subsection we are focusing more on the problem of finding unwanted regularities in a traffic flow to identify ongoing attacks.

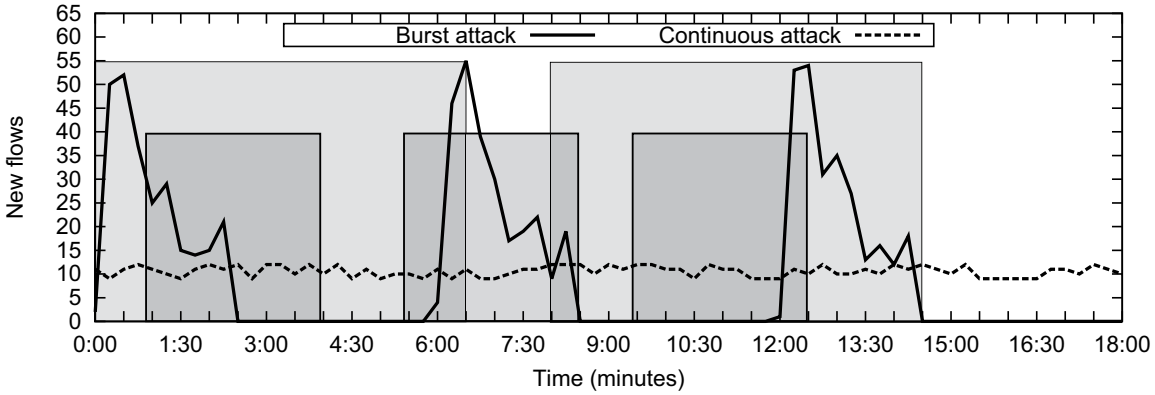


Fig. 1. Visualization of two types of attacks and two possible time window settings.

A Fourier transform (FT) basically generates a frequency spectrum of a given signal, i.e. decomposes the signal into a set of simple oscillating functions. The results of a FT show which frequency components are dominant in the signal. Figure 2 shows the result for the burst and the continuous attacks. Peaks in the burst attack transformation point to a presence of a dominant repeating pattern, i. e. the attack. On the other hand, a FT of a continuous attack does not reveal anything and is akin to a FT of non-malicious traffic.

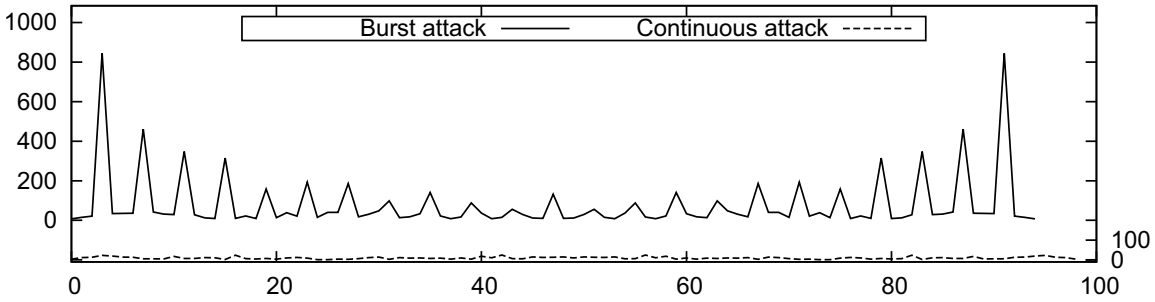


Fig. 2. Fourier transform of attacks in the Figure 1. *Burst attack is offset 200 points up the y-axis for clarity.*

Discussion Both, the window heuristic and the Fourier transform allow the discovery of almost constant periodic traffic that may be a symptom of an ongoing attack. Each method fills a specific niche; the heuristic is more useful for evenly distributed attacks, the Fourier transform for attacks with more complicated patterns.

2.4 Detecting Abrupt Changes in Entropy Time Series

In [6], entropy was proposed as a metric to discover a wide range of anomalies in flow traces. Entropy can reduce high-dimensional network traffic to a single numeric value. While this simplification inevitably implies a loss of information, it has the benefit of reduced computational complexity. The use of entropy makes it possible to discover various occurrences of brute-force attacks, such as (D)DoS, large-scale scanning activity and worm outbreaks in the observed flow traces since these types of attack heavily modify the distribution of the underlying flow features.

Method In [11], we proposed to use the normalized Shannon entropy to form time series for five flow features: Source and destination IP address, source and destination port and packets per flow. The normalized Shannon entropy normalizes all entropy values to the interval $[0,\ 1]$ so that all five time series become directly comparable.

Every data point for the resulting five time series is computed by calculating the entropy over a sliding window which consists of all observed flows within the last five minutes. In addition, the sliding windows overlap by four minutes. The overlap should lead to more fine grained results at the cost of being computationally slightly more expensive.

Figure 3 illustrates an example. The x-axis depicts time whereas the y-axis shows the normalized Shannon entropy ranging from 0 to 1. The spike in all time series at May 26 at around 7am could be considered an abrupt change which should be detected by the proposed algorithm.

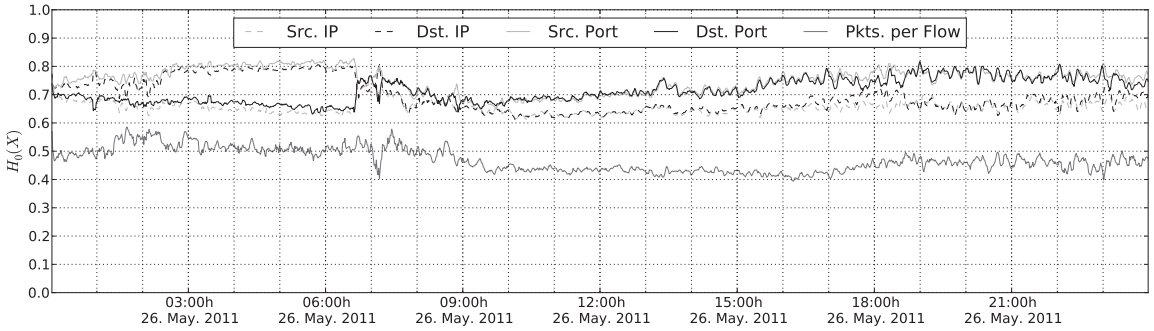


Fig. 3. Entropy time series for five flow features.

The main idea of the proposed algorithm is to continuously conduct *short-term predictions* about the progression of the respective time series. The simple exponential smoothing algorithm is used for predictions. After every prediction, the difference to the measured entropy value, the so-called prediction error, is computed.

The higher this difference, the more abrupt can the change in the time series be considered.

However, the prediction errors are not equally significant since the underlying entropy time series exhibit different statistical distributions. To make all five time series equally significant, we multiply them with a weight factor which is calculated based on the empirical standard deviation of the respective time series. Finally, all five prediction errors are summed up to a single anomaly score. This makes it possible to configure a single threshold. As soon as the anomaly score exceeds the configured threshold, an alert can be fired.

Discussion The evaluation and real-world experiments conducted in [11] suggest that the algorithm scales very well. In addition, the approach is easy and fast to deploy.

3 Flow Limitations

The previous sections showed how it is possible to use network flows to discover anomalies and intrusions. Using network flows for attack and anomaly detection also has its downsides which are discussed below.

3.1 Information Loss

When network traffic is converted to network flows, certain information remains (IP addresses and ports) and some information is derived (bytes per flow or packets per flow). Other information, like the packet payload, is inevitably lost. This loss of information implies that network flows are not suitable for the detection of all kinds of malicious network activity. This is not an issue in case of flow-based brute-force attack detection, but attacks which manifest solely in packet payload, such as remote exploits, are virtually invisible in network flows.

3.2 Collection Delays

Flow-based detection does not happen in real-time because flow exports are generally driven by three timeouts. The *active* timeout splits long-lasting flows and the *passive* timeout exports slow flows. These two timeouts influence flow aggregation considerably and therefore the detection based on volume characteristics. The third timeout is used for periodically sending acquired flows from the metering to the collecting process. Traditionally, the size of this time window is 5 minutes. That means, the acquired flows are sent at least with a 5-minute delay which could be considered too late in case of attack which happen very fast.

3.3 Packet Sampling

In [1], Brauckhoff et al. evaluated the influence of packet sampling on anomaly detection. The authors made use of an unsampled data set containing activity of the Blaster worm. Using a traffic baseline which lacks the activity of Blaster, the authors simulated packet sampling at increasing rates and could thus estimate the impact of packet sampling.

Their findings suggest that with increasing sampling rates, the number of bytes or packets is still useful to detect Blaster activity. The number of flows is strongly biased by sampling, though. Besides, the authors conclude that entropy-based flow summarization is less affected by packet sampling than volume-based summarization such as bytes or packets per flow.

4 Detection Evasion

The methods proposed in Section 2 operate with several assumptions and constraints that dictate what types of attack each method can detect. If these constraints are taken into account by an attacker, it is possible to come up with relatively straightforward methods of detection evasion that can severely limit the potential of proposed methods.

4.1 Attack Function Separation

As was described in [4], the SSH attack typically has three phases: Scanning, Brute-force and Die-off. In our experience, this scheme is not limited to SSH attacks and can be applied to other protocols and authentication schemes. Flow-wise, the scanning phase is identified by large amount of small flows destined to a large number of targets. There are usually lots of unsuccessful connections to closed or filtered ports. In the brute-force phase, recorded flows are larger and destined to specific targets with virtually no unsuccessful connections. The die-off phase that is equal to a successful attack has usually few large and infrequent flows.

Both, [9] and [4] expect to some extent a specific attacking machine to go from Scanning phase over Brute-force phase and eventually to Die-off phase. This is a valid heuristic for attackers with limited number of machines at their disposal. However, attackers with more resources can allocate their machines into groups specialized on each phase, thus evading this heuristic with only a little overhead.

4.2 Hiding Under Threshold

Since the flow-based detection methods cannot inspect data in the same detail as deep packet inspection, they are inherently dependent on using thresholds to differentiate between normal and deviant behavior. These thresholds are usually

tailored towards a given network and either determined manually or automatically. Looking at some methods in recent papers, e.g., [7], it is apparent that the general approach is to use conservatively high thresholds to avoid false positives. These methods are suited for noisy attacks and can be subverted by limiting the attack intensity. Attackers can thus stretch their attacks in both, time and volume in order to stay under the respective thresholds and evade detection.

4.3 Temporal Distortions

Commonly available brute-forcing tools (e.g., AccessDiver, Sentry, Hydra) allow attackers to configure the number of threads that will run in parallel. This effectively determines the average amount of attack attempts per given time window. This amount remains relatively constant during the course of an attack. This behavior is exploited by methods presented in Section 2.3 to detect attacking machines.

By introducing temporal distortions (e. g., random delays) to traffic flows, the attacker can disrupt the periodicity and the automated profile of an attack, thus rendering the detection method ineffective. The NCrack brute-forcer is going partly this way by adapting its brute-forcing process automatically to network conditions (e.g., waiting with the next attack attempts after blocking of a machine or proxy), but to the authors' knowledge, there is no available brute-forcer to employ this relatively cheap and easy technique.

4.4 Flow Stretching

Some detection methods that were discussed so far relied on particular flows being relatively short and only a few packets in volume. This is reasonable, given that attacking machines usually exchange the bare minimum of data in order to try to authenticate. Flows of a particular attack thus look alike; the only difference is a few bytes, usually influenced by username and password length. There is, however, danger in such an approach. Several important protocols — SSH, RDP and HTTP(S) — by design allow the exchange of arbitrary data in the process of authentication, before the final decision whether to let an attacker in or not is made. This arbitrary data can be used to inflate both the flow size and the flow duration. Such inflated flows no longer look alike. Instead, they can be made to look like valid communication and avoid detection.

5 Summary

This chapter gave an overview about current research in the field of flow-based attack and anomaly detection. We summarized state of the art concepts for attack detection, especially brute-force ones, and concluded the chapter by discussing evasion strategies as well as the limitations inherent to the process of detecting attacks in network flows.

References

1. D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of Packet Sampling on Anomaly Detection Metrics. In *Internet Measurement Conference*, pages 159–164, Rio de Janeiro, Brazil, 2006. ACM.
2. C. Callegari, M. Pagano, S. Giordano, and T. Pepe. Combining wavelet analysis and information theory for network anomaly detection. In *International Symposium on Applied Sciences in Biomedical and Communication Technologies*, pages 1–5, Barcelona, Spain, 2011. ACM.
3. P. Haag. NFDUMP. `http://nfdump.sourceforge.net/`.
4. L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. In *International Conference on Autonomous Infrastructure, Management, and Security*, Luxembourg, Luxembourg, 2012. Springer.
5. C.-T. Huang, S. Thareja, and Y.-J. Shin. Wavelet-based Real Time Detection of Network Traffic Anomalies. In *SecureComm and Workshops*, pages 1–7, 2006.
6. A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. *SIGCOMM Comput. Commun. Rev.*, 35(4):217–228, 2005.
7. M. Rehak, M. Pechoucek, P. Celeda, J. Novotny, and P. Minarik. CAMNEP: agent-based network intrusion detection system. In *International Conference on Autonomous agents and multiagent systems*, pages 133–136, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.
8. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-Based Intrusion Detection. *Communications Surveys Tutorials, IEEE*, 12(3):343 –356, quarter 2010.
9. J. Vykopal. A Flow-Level Taxonomy and Prevalence of Brute Force Attacks. In *Advances in Computing and Communications*, pages 666–675, Kochi, India, 2011. Springer.
10. J. Vykopal, T. Plesnik, and M. Pavel. Network-Based Dictionary Attack Detection. In *International Conference on Future Networks*, pages 23–27, Bangkok, Thailand, 2009. IEEE Computer Society.
11. P. Winter, H. Lampesberger, M. Zeilinger, and E. Hermann. On Detecting Abrupt Changes in Network Entropy Time Series. In *Communications and Multimedia Security*, pages 194–205, Ghent, Belgium, 2011. Springer.

Malware in Hardware Infrastructure Components

Christian Krieg and Edgar Weippl

SBA Research,
Favoritenstraße 16, 1040 Vienna, Austria
{ckrieg,eweippl}@sba-research.org

Abstract. Malicious hardware is a fairly new research topic that has attracted the interest of the scientific community. Therefore, numerous approaches have been proposed in the last years to counter the threat of so-called *hardware Trojans*. This chapter describes malicious hardware in the context of the security of hardware infrastructure components. Network infrastructure plays a vital role in our everyday lives, since many services depend on reliable and secure connections. In the following, we briefly introduce the topic of hardware Trojans. After describing their basic components, we give some insights into how hardware Trojans can be used maliciously in infrastructure devices. Furthermore, we outline the evolution of hardware Trojans and measures to counter them.

1 Introduction

Hardware infrastructure components are becoming increasingly vital in our everyday lives. Sensitive data, such as credit card numbers, medical data, private communications, and banking information are transmitted over communication channels, in most cases without the users being aware of it. It is, therefore, of utmost relevance that communication channels are secure against adversaries who try to gain access to sensitive data. Encryption and security protocols aim to secure communication from a sending device to a receiving device. Cryptography is applied within these devices, which means that data are available in plaintext. Moreover, cryptographic keys are available in such devices, which make them a potential point of interest for attackers who aim to compromise data confidentiality. One way to gain knowledge of cryptographic keys would be to tamper with the device in a way that allows the attacker to read out the keys from memory [5]. Other possibilities include fault injection [25] combined with analysis methods such as side-channel analysis [25]. In this contribution, we will consider a fairly new attack vector against the security of hardware infrastructure components. We will examine how malicious hardware inclusions – so-called *hardware Trojans* – affect the overall security of infrastructure components, focusing on data confidentiality. We outline the evolution of hardware Trojans and countermeasures in the literature.

When talking about malware in hardware infrastructure components, we focus on functions of a system that are implemented in hardware but have not been specified. It is the hardware itself that is malicious.

Malware can be introduced into hardware in many ways. For example, a malicious designer can inject an unspecified functionality into a design by adding just

a few lines of hardware description code [45]. Furthermore, a synthesis tool can be modified by adversaries is that the hardware to be synthesized is altered in an unspecified manner [41]. And even if it is expensive and resource-intensive, a malicious chip producer can reverse-engineer a design to include circuitry at the physical level [2].

Hardware Trojans will be the topic of this chapter. Section 2 will briefly classify hardware Trojans. Section 3 presents some ideas on how malicious hardware can subvert an infrastructure and how sensitive data can be leaked by hardware Trojans. In Section 4, we outline the historical development of hardware Trojans, always being aware of defense methods to counter Trojans.

2 Components of Hardware Trojans

As hardware Trojans have to pass functional tests without being detected, they have two basic mechanisms: a *trigger* and a *payload* mechanism [50].

A trigger should activate the payload upon a certain condition, such as the occurrence of a rare event (e.g., a bit pattern 0x3745 on a data line), the lapse of a certain time interval (e.g., 10,000 seconds), or a condition that is met by the environment (e.g., temperature is 65°C). The most essential requirement of a trigger condition is that it is not met during functional tests, which are integral parts of the hardware production process. Otherwise, it could activate the Trojan during the test, which would make it more detectable.

The payload mechanism implements the effective function of a Trojan. For instance, this could be a kill switch (i.e., the permanent deactivation of a hardware system), the interception of sensitive data such as a cryptographic key, or the remote control of a hardware system (which corresponds to opening a hardware backdoor).

3 Hardware Trojans in Infrastructure Components

Section 4 provides a comprehensive overview of the approaches to Trojan detection. It shows that the problem is manifold and illustrates the wide range of attack vectors.

Although there have been no reports of an actual attack so far, recent research has addressed many threats to infrastructure components. For example, Jin et al. show that it is possible to leak the cryptographic key of a wireless device over the wireless channel [23]. Depending on each bit of the key, the wireless signal is altered within the permitted tolerances. That way, an attacker only has to reside within the range of the wireless device, record the signal, and perform a statistical analysis to obtain the key. Subsequently, the attacker can use the key for authentication and use the device as legitimate user, which enables her to subvert the entire infrastructure of which the device is a part.

Similarly, Lin et al. show how data can be leaked over a secret channel [32]. By modulating the power supply signal of a device, sensitive data can be leaked stealthily. It is hard to detect the covert transmission of data, since the signal is modulated in the code division multiplex, i.e., spread spectrum technology is used. Therefore, without knowledge of the correct code, the covert signal cannot be detected, as it is indistinguishable from noise. To obtain sensitive data (such as a cryptographic key), an attacker has to probe the power supply signal of the device under attack and demodulate it by correlating it with the proper code.

Likewise, King et al. designed a malicious processor, which has hidden hardware implemented that allows an attacker to perform extensive attacks on the software layer [27]. The *Illinois Malicious Processor* provides mechanisms to illegitimately login to an overlying operating system as administrative user without providing a password. This way, an attacker can get broad access over an infrastructure component. If such a processor is deployed, e.g., in a router, the infrastructure itself could be modified, therefore serving as a base for further attacks on the network layer.

4 Evolution of Hardware Trojans and Their Countermeasures

Hardware Trojans have become a serious problem for the security of IT systems. In the following, we outline the historical evolution of Trojans, which is accompanied by the evolution of countermeasures to fight Trojans. We describe the development from 2005, when the US Department of Defense published a report on the supply of semiconductors [19], to the present. To maintain logical association, we have grouped the approaches into subsections.

4.1 Raising Political Awareness

In 2005, the US Department of Defense released a report about the security of supply of high-performance integrated circuits [19]. In this report, they investigated the concept of a vertical business model for compliance with the demand for secure and authentic hardware. The report stated that the manufacturing of microchips had been relocated to low-wage countries for financial reasons. Therefore, the risk that chip manufacturers could add additional functions during the production process appeared possible. The report described trustworthiness as follows:

> "*Trustworthiness* includes confidence that classified or mission critical information contained in chip designs is not compromised, *reliability* is not degraded or unintended design elements inserted in chips as a result of design or fabrication in conditions open to adversary agents. Trust cannot be added to integrated circuits after fabrication; electrical testing and *reverse*

engineering cannot be relied upon to detect undesired alterations in military integrated circuits." [1]

As the production consistently had been transferred to potential enemies, the US Department of Defense did not believe that the supply of necessary semiconductors could be ensured in the event of war.

For this reason, the research project TRUST in Integrated Circuits (TIC) was started in 2007 by the Defense Advanced Research Projects Agency (DARPA) [17]. TIC is intended to develop technologies that can provide trust for circuits in the absence of a trusted foundry. It only considers technical efforts that address the fabrication of Application-Specific Integrated Circuits (ASICs) by non-trusted foundries and software implementation of configurable hardware, such as Field-Programmable Gate Arrays (FPGAs).

[1] triggered a veritable flood of publications with his article on the threat of hardware Trojans.

2008 can be clearly identified as the year in which this topic gained academic interest. The Australian Department of Defence picked up the topic and published a report about the battle against hardware Trojans, evaluating its effectiveness [4].

4.2 Introducing Side-Channel Analysis

In the same year, [2] published their work on a method to detect secretly added functionality through side-channel analysis. A device under test is monitored with regard to physical side channels, such as supply voltage or timing. This work can definitely be seen as the starting point for numerous publications on this topic.

The focus at the time was clearly on side-channel analysis [7,10,11,24,31,38,47], but other methods in the area of logic tests [15] were proposed as well. To augment the detection rate, some approaches for increasing the activation of hardware Trojans were proposed [40,22].

4.3 Malicious Computer Systems

[27] were the first to publish a comprehensive combined hardware/software attack. In this attack, a hardware Trojan serves as the basis of an extensive attack by allowing an attacker to sign on to the operating system with root privileges with the help of a hardware backdoor.

The University of New York held a competition where the goal was to implement hardware Trojans. The criteria for the competition were to insert malicious circuits into the original layout as unnoticeably as possible; the extraction of information without being noticed was also part of the position in the final ranking. Work submitted to the competition can be found in [16,12].

[1] [19], p. 3

4.4 Increasing Trojan Activation

Many approaches have been presented for increasing the chance for Trojan activation, which should help improve the detection rate during functional tests. *Toggle minimization* is used to reduce the overall activity of a circuit to be able to measure the (partial) activity of a Trojan, if one is present [9].

Inverting the supply voltage of the logic gates in a circuit causes the logic states of the gates to be inverted as well. This measure causes an inversion of the detectability – a Trojan that was previously hard to spot can now be easy to detect [8].

Generating optimal testing patterns should increase the detection rate for logic tests. [14] present an approach for initiating rare logic states multiple times in order to help trigger a potential trigger condition. Rare states are identified by a statistical method.

[42] increase the chance of state changes ("toggles") by inserting dummy flip-flops into the original design. Dummy flip-flops are realized as scan flip-flops to preserve the original functionality.

[6] present an approach that first determines signals that are easy to activate during a functional test. These signals are then ignored when testing for the presence of Trojans that are hard to identify. Using the remaining signals, a formal verification is carried out. Any detected Trojans are subsequently isolated.

4.5 Applying Gate-Level Characterization to Trojan Detection

Generally, side-channel analysis should detect deviations from expected behavior caused by hardware Trojans. Because Trojans strive to be hard to detect during a functional test, the impact of a Trojan is assumed to be as small as possible compared to overall circuit activity.

This is a problem for their detection, because the effect of process variations will be of almost the same order of magnitude as Trojan impact.

The approach of GLC (Gate-Level Characterisation) tries to characterize each gate of an IC. Here, the values performance, switching power and leakage current are used for characterization. Scaling factors are calculated to account for process variations that cannot be avoided in the manufacturing process. During a functional test, scaling factors are measured with the help of side-channel analysis. If the testing results of an IC differ too much from the calculated characteristics, a Trojan may have been implemented [36,37].

4.6 Proposing Trojan-Resistant Bus Architectures

Trojans that have been inserted into complex hardware can also be detected with the help of the operating system. [13] propose an approach where a simple hardware guard monitors accesses from the CPU to the memory data bus and performs

liveness checks. A watchdog timer is started each time the monitor observes a certain pseudorandom memory access procedure, which is initiated by the operating system. If the watchdog times out, a Denial-of-Service (DoS)-attack is detected. The operating system also periodically checks if memory protection is activated to prevent privilege escalation attacks.

[26] propose a Trojan-resistant bus architecture for Systems-on-Chip (SoCs). The architecture is able to detect unauthorized bus accesses. To prevent DoS attacks, permanent bus allocation to one bus node is blocked by limiting maximum bus allocation time.

4.7 Stealthy Trojan Communication

A very interesting new class of Trojans is introduced in [32], where they present a new technology called *Malicious Off-Chip Leakage Enabled by Side-Channels (MOLES)*, which allows the extraction of sensitive data with the help of spread-spectrum technology. Because the signal of the extracted information completely disappears in noise, a detection of the hidden data transfer is hardly ever possible. [33] describe how data can be transmitted by modulating the power supply signal using spread-spectrum techniques. The implementation makes use of high capacitances that draw current while charging. Depending on whether a zero or one is to be transmitted, a capacitor is charged or not. The charging current – encoded via spread-spectrum technique – can be analyzed by performing a side-channel analysis on the power supply.

4.8 Securing Multi-Core Architectures

Another approach to detect Trojans in multi-core systems is proposed by [34]. In this approach, software to be executed is varied while keeping functional equivalence. This can be accomplished by using different compilations or alternative algorithms. The variants of the software are executed on multiple cores. If one variant of the software matches the trigger condition of an injected Trojan – thus activating it – the results of two calculations will differ. This way, a Trojan can be detected and isolated at runtime.

4.9 Introducing Run-Time Detection

The *BlueChip* approach introduced by [21] relies on additional hardware modules. It is designed to render hardware Trojans injected at design time harmless at runtime.

Trojans are isolated by replacing suspicious circuits by software emulation. Suspicious circuits are identified with Unused Circuit Identification (UCI), which is a method that monitors the activity of a circuit during the functional test. If a part of a circuit remains unused during the entire testing period, it is considered to be

assigned to a Trojan circuit (which should not be detected during functional tests and would, therefore, remain silent).

[45] present an approach to combat Trojans especially in microprocessors at runtime. The assumption is that malicious functionality is injected during the design phase by malicious designers. The following constraints are defined:

1. The number of malicious designers is low,
2. the activity of malicious designers remains unnoticed,
3. the attackers need few resources to inject a backdoor,
4. the backdoor is activated by a trigger,
5. the Read-Only Memories (ROMs) written during the design phase contain correct data (microcode).

Two types of backdoors serve as a Trojan model: *emitter* (send data) and *corruptor* backdoors (modify data). The latter are very difficult to detect, since their operations can be hard to distinguish from normal, legitimate operations. The proposed measure to prevent the hardware backdoors is an on-chip monitoring system that consists of four parts: predictor, reactor, target and monitor.

A Trojan is discovered if the result of the monitored unit does not match the predicted outcome of the predictor. The detection principle is based on the assumption that the monitored unit never communicates with the monitoring instance – therefore, the designer of a malicious unit X cannot corrupt the monitor of X.

4.10 Improving Side-Channel Analysis

[20] propose another approach based on side-channel analysis to detect hardware Trojans. The power consumption of a particular region of an IC is compared to the power consumption of the same region of another IC. If the power consumptions differ greatly, the reason could be a Trojan. This technique is called *self referencing*.

[35] partition a layout where the different partitions are stimulated through appropriate test patterns. Transient current (I_{DDT}) und maximum frequency (f_{max}) are determined via side-channel analysis. Because I_{DDT} and f_{max} are linearly dependent and f_{max} is unalterable, a Trojan can be detected by observing an increase of I_{DDT}.

To determine the smallest detectable Trojan, [39] examine the sensitivity of a transient analysis of power supply signals. The smallest detectable Trojan consists of a single logic gate under laboratory conditions if the Trojan responds to a test pattern. If the measurement is characterized by a Signal-to-Noise-Ratio (SNR) of 10 dB, the size of the smallest recognizable Trojan increases to seven gates.

4.11 Trojan Localization

[43] rearrange scan chains to increase the detection of hardware Trojans. The appliance used to test integrated circuits is usually not bound to a specific layout on

the chip. The approach suggests a layout that rearranges scan chains over the entire chip area, so that certain areas can be specifically activated or deactivated during the functional test.

This should make the activity of a Trojan visible. Experimental tests show a partial amplification of Trojan activity by a factor of 30.

4.12 Enhanced Gate-Level Characterization

[49] introduce a way of detecting Trojans using the method of GLC with thermal conditioning. Thermal conditioning means that an IC is intentionally heated unevenly. This method exploits the fact that the leakage power increases exponentially with temperature.

The aim is to eliminate correlations when measuring leakage power that are caused by dependencies of gates with other gates. By heating correlated gates differently, more variability is introduced into the calculation results. The entire process is calculated using a simulation model. Simulation results can be used to obtain scaling factors in order to calibrate the measurement procedure to minimize measurement variations caused by process variations. The advantage of applying this method is the full characterization of all gates of an IC.

The detection of Trojans by GLC with thermal conditioning is not suitable for large circuits, because properties are determined for the entire circuit. Attackers can exploit this fact and inject ultra-small Trojans, whose impact will disappear in measurement noise [48]. To make the process scalable and thus useful for the analysis of large ICs, [48] extend the process by adding a preceding step of segmentation, which decomposes a large circuit into many small sub-circuits.

The segmentation criteria are chosen in a way that ensures that the results of the following GLC are as accurate as possible. The segmentation process itself is accomplished by varying an amount of primary input vectors and, at the same time, 'freezing' the other input vectors. The circuit part obtained by segmentation is now seen as an independent part of the circuit (i.e., a segment). A GLC with thermal conditioning, which is applied to this segment, provides information about the presence of Trojan. A subsequent identification mechanism based on the principle of 'guess and verify' should provide information about the type and the input pins of any existing Trojan.

4.13 Data Leakage by Trojans

[23] present an attack that aims to leak an Advanced Encryption Standard (AES) key. This is achieved by manipulating the transmission signal of a wireless link within its tolerances. This work is the first presented attack in the analog domain.

Using an external guardian core, [18] propose an approach to prevent data leaks via the data bus caused by hardware Trojans. The watchdog monitors the access

behavior to the main memory by comparing each memory access with an emulated version of it. If the memory accesses match, it is approved by the guard, otherwise it is discarded. The emulation of the memory accesses is done in the software applications that are executed on the system.

4.14 Defining Threat Models

A new class of attacks against cryptographic algorithms is presented by [3]. These so-called multi-level attacks are based on the interaction of multiple people involved in the hardware design and production process.

The authors present an example of such an attack, where the secret key of a hardware implementation of the AES algorithm is leaked through a hidden channel of the power supply.

The authors assume a link between the developer and the operator of cryptographic hardware. The developer inserts the malicious circuit into the original circuit. After manufacturing, the secret key can be read by the operator. Collaboration between the two parties is necessary because otherwise, without knowledge of the technology used, the operator would not be able to read the key.

4.15 Combining Different Approaches to Side-Channel Analysis

A first approach for combining the possibilities of different methods of side-channel analysis is presented by [28], which is an advancement of [29].

The framework allows analysis by different side channels and the use of different evaluation methods, such as quiescent current, leakage current, and delay.

The mathematical analysis of the measurement results is based on GLC and a subsequent statistical analysis. Here, a new objective function is defined for the linear program that takes into account the *submodularity* of the problem: The impact of a Trojan on a side channel is greater the smaller the analyzed circuit is.

After GLC, the deviation of the measurement results (obtained by the various side-channel analyses) from the expected values is calculated for each gate. A sensitivity analysis is performed so that possible malicious circuits can be detected. The design of the Trojan determines the effect on the side channels. Some Trojans are more likely to have an impact on power consumption, while others will affect the performance. The measurement results of the different analyses (*multimodal*) are combined to achieve a higher detection rate. Experiments show that if the Trojans are inserted into areas with adequate sensitivity, the detection rate is 100%. The reverse is true as well: This method can also be used to find areas in which Trojans are most difficult to detect.

[30] also evaluate the effectiveness of the combined results from the analyses of various side channels. In contrast to [28], however, they present no general model in which the results of different side-channel analyses could be combined. They show

that by combining transient power and performance, followed by regression analysis, higher detection rates can be achieved in contrast to using each side channel on its own. Experiments show detection rates of up to 80% when they are not calibrated. After calibration, a detection rate of 100% can be achieved.

4.16 Increase Trojan Activation by Additional Flip-Flops

To increase the probabilities of state transitions in circuits during functional tests, [44,42] present an approach to insert dummy scan flip-flops into the original circuit layout. The consideration is that Trojans will be activated completely or in part and, thereby, have an impact on side channels. For example, the gates of a Trojan could switch – the additional energy consumption could be visible in the course of a side-channel analysis. The biggest motivation is to shorten the authentication process of an IC.

The method is carried out as follows: first, a threshold for the switching probability is determined that takes into account technical and economic considerations. Then, the switching probabilities of each sub-net are determined and the nets are divided into two groups, one with high and one with low switching probabilities. A dummy scan flip-flop is connected to nets with low switching probability to increase the probability of a transition. A net that tends to logic 0 is succeeded by a scan flip-flop that draws the output of the net (in a testing setup) to logic 1 if needed. The opposite is true for nets that tend to logic 1.

4.17 Avoiding Trojans

[46] present an approach to avoid Trojans which is intended to prevent the occurrence of the trigger condition of digital, deterministic triggers.

Untrusted data is not monitored and manipulated within functional groups, but rather on their inputs and outputs. The idea is that data is scrambled and obscured in a controlled manner, so that a trigger cannot find a valid trigger condition and, therefore, a Trojan will never become active. The following types of triggers are considered:

1. ticking timebomb,
2. single-shot cheat code and
3. sequence cheat code.

A ticking timebomb is a time-controlled trigger that is activated when a predetermined number of clock cycles N has elapsed. The clock cycles that have already passed are usually determined using a counter. If this counter is always reset before reaching state N, a Trojan will never become active. This can be achieved by periodically resetting the entire digital system (*power reset*). The reset interval must

be smaller than the testing period T that the obligatory functional test requires. If an attacker wants to achieve that a ticking timebomb will be activated, this must occur within N clock cycles. However, if $N < T$, the Trojan will be activated and detected during the functional tests.

Data-based triggers can be separated into single-shot cheat code and sequence cheat code triggers. Single-shot cheat code triggers are activated when a certain rare value is applied to the monitored interface. To prevent a rare value from being knowingly applied to an input of a compromised functional unit by an attacker, it is obfuscated by encryption in such a way that it no longer meets the trigger condition. Simple encryption methods are, for example, XOR, PUF, or random values. This approach is valid for non-computational units such as memory, etc. In order to protect computational units (e.g., ALU), *homomorphic* functions are used. Homomorphic functions obey the following rule: $f(g(x), g(y)) = g(f(x, y))$. An example of a homomorphic function is: $x^2y^2 = (xy)^2$. If we assume that the computational function is the squaring, a non-trustworthy value x that is to be processed is multiplied by a random value y before it is squared. To obtain a valid result, the result on the output of the functional unit must be divided by y^2.

The last class of triggers, sequence cheat codes, is countered by scrambling and inserting dummy loads. The scrambling is achieved by simple reordering. If this is not possible, dummy loads can be inserted into the data stream. A maximum number n of bits must be defined which is then allowed as a valid sequence. After n processed bits, a dummy load is inserted to avoid execution.

4.18 Using Ring Oscillators for Trojan Detection

[51] use a network of ring oscillators to detect injected Trojans. A ring oscillator is a simple circuit for generating oscillations, consisting of an odd number of similar gates.

The principle of detection is based on the fact that the frequency of a ring oscillator is influenced by physical parameters. Accordingly, the frequency also depends on the supply voltage V_{DD}. If V_{DD} drops, the propagation delay of the gates increases. This in turn means that the delay of the entire ring oscillator and, therefore, its cycle duration increases, which is equivalent to a drop in frequency.

A drop in V_{DD} occurs when a gate draws current. If the use of CMOS is assumed, this is the case with each switch of a transistor of a gate, hence, with every state change. If a Trojan is implemented in an IC, adjacent ring oscillators will see a stronger decrease in V_{DD} and, consequently, a frequency drop compared to ICs without a Trojan.

To achieve the best possible coverage, ring oscillators are distributed over the entire chip area. Using statistical methods, in which the frequencies of the built-in ring oscillators are evaluated (simple outlier analysis, principal component analysis and advanced outlier analysis), a detection rate of 100% is achieved. The validation

using an FPGA provides detection rates between 80% and 100%. The robustness of the approach against direct attacks is considered very high, because the manipulations have direct impact on the frequency of the ring oscillators and, therefore, become visible immediately in functional tests.

5 Conclusion

Malicious hardware presents a serious threat to infrastructure components. Taking into account that powerful embedded systems, such as smartphones, are widely used today, a gloomy picture can be painted for the future. As they are extensively connected to the internet, a broad class of targets is available.

In this chapter, we showed how malicious circuits could be used to subvert an infrastructure by taking over its components. Further, we presented the basic components of hardware Trojans, i.e., trigger and payload mechanisms. We also outlined how hardware Trojans have evolved over time, as well as the methods to counter them.

References

1. S. Adee. The Hunt For The Kill Switch. *Spectrum, IEEE*, 45(5):34 –39, may. 2008. ISSN 0018-9235. doi: 10.1109/MSPEC.2008.4505310.
2. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan Detection using IC Fingerprinting. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 296 –310, may. 2007. doi: 10.1109/SP.2007.36.
3. Sk. Subidh Ali, Rajat Subhra Chakraborty, Debdeep Mukhopadhyay, and Swarup Bhunia. Multi-level attacks: An emerging security concern for cryptographic hardware. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pages 1–4, 2011. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5763307`.
4. M. S. Anderson, C. J. G. North, and K. K. Yiu. Towards Countering the Rise of the Silicon Trojan. Technical report, 12 2008. URL `http://dspace.dsto.defence.gov.au/dspace/bitstream/1947/9736/1/DSTO-TR-2220%20PR.pdf`.
5. Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001. ISBN 0471389226. URL `http://www.cl.cam.ac.uk/~rja14/Papers/SE-14.pdf`.
6. M. Banga and M. S. Hsiao. Trusted RTL: Trojan detection methodology in pre-silicon designs. In *Proc. IEEE Int Hardware-Oriented Security and Trust (HOST) Symp*, pages 56–59, 2010. doi: 10.1109/HST.2010.5513114.
7. M. Banga and M.S. Hsiao. A region based approach for the identification of hardware Trojans. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 40 –47, jun. 2008. doi: 10.1109/HST.2008.4559047.
8. M. Banga and M.S. Hsiao. VITAMIN: Voltage inversion technique to ascertain malicious insertions in ICs. In *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, pages 104 –107, 2009a. doi: 10.1109/HST.2009.5224960.
9. M. Banga and M.S. Hsiao. A Novel Sustained Vector Technique for the Detection of Hardware Trojans. In *VLSI Design, 2009 22nd International Conference on*, pages 327 –332, 2009b. doi: 10.1109/VLSI.Design.2009.22.

10. Mainak Banga. Partition based Approaches for the Isolation and Detection of Embedded Trojans in ICs. Master's thesis, Faculty of Virginia Polytechnic Institute and State University, 09 2008. URL http://scholar.lib.vt.edu/theses/available/etd-09042008-155719/unrestricted/MS_Thesis_Mainak.pdf.
11. Mainak Banga, Maheshwar Chandrasekar, Lei Fang, and Michael S. Hsiao. Guided Test Generation for Isolation and Detection of Embedded Trojans in ICs. In *GLSVLSI '08: Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pages 363–366, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-999-9. doi: http://doi.acm.org/10.1145/1366110.1366196.
12. Alex Baumgarten, Michael Steffen, Matthew Clausman, and Joseph Zambreno. A case study in hardware Trojan design and implementation. *International Journal of Information Security*, 10:1–14, 2010. ISSN 1615-5262. URL http://dx.doi.org/10.1007/s10207-010-0115-0.
13. G. Bloom, R. Simha, and B. Narahari. OS support for detecting Trojan circuit attacks. In *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, pages 100 –103, 2009. doi: 10.1109/HST.2009.5224959.
14. Rajat Chakraborty, Francis Wolff, Somnath Paul, Christos Papachristou, and Swarup Bhunia. MERO: A Statistical Approach for Hardware Trojan Detection. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 396–410. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-04138-9_28.
15. R.S. Chakraborty, S. Paul, and S. Bhunia. On-demand transparency for improving hardware Trojan detectability. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 48 –50, jun. 2008. doi: 10.1109/HST.2008.4559048.
16. Z. Chen, X. Guo, A. Nagesh, M. Reddy, and A. Maiti. Hardware Trojan Designs on BASYS FPGA Board. http://filebox.vt.edu/users/xuguo/homepage/publications/csaw08.pdf, 2008. URL http://filebox.vt.edu/users/xuguo/homepage/publications/csaw08.pdf.
17. DARPA. Trust in Integrated circuits (TIC). http://www.darpa.mil/MTO/solicitations/baa07-24/index.html, Mar 2007. URL http://www.darpa.mil/MTO/solicitations/baa07-24/index.html.
18. A. Das, G. Memik, J. Zambreno, and A. Choudhary. Detecting/preventing information leakage on the memory bus due to malicious hardware. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 861 –866, mar. 2010. URL http://portal.acm.org/citation.cfm?id=1871135.
19. Defense Science Board, Department of Defense, U.S.A. High Performance Microchip supply. http://www.cra.org/govaffairs/images/2005-02-HPMS_Report_Final.pdf, 02 2005. URL http://www.cra.org/govaffairs/images/2005-02-HPMS_Report_Final.pdf.
20. Dongdong Du, Seetharam Narasimhan, Rajat Chakraborty, and Swarup Bhunia. Self-referencing: A Scalable Side-Channel Approach for Hardware Trojan Detection. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-15031-9_12.
21. Matthew Hicks, Murph Finnicum, Samuel T. King, Milo M. K. Martin, and Jonathan M. Smith. Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 159 –172, May 2010. doi: 10.1109/SP.2010.18.
22. S. Jha and S.K. Jha. Randomization Based Probabilistic Approach to Detect Trojan Circuits. In *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*, pages 117 –124, 2008. doi: 10.1109/HASE.2008.37.
23. Y. Jin and Y. Makris. Hardware Trojans in Wireless Cryptographic ICs. *Design Test of Computers, IEEE*, 27(1):26 –35, jan. 2010. ISSN 0740-7475. doi: 10.1109/MDT.2010.21.
24. Yier Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 51 –57, 2008. doi: 10.1109/HST.2008.4559049.
25. Chong Hee Kim and J.-J. Quisquater. Faults, injection methods, and fault attacks. *IEEE Design & Test of Computers*, 24(6):544–545, 2007. doi: 10.1109/MDT.2007.186.

26. Lok-Won Kim, J.D. Villasenor, and C.K. Koc. A Trojan-resistant system-on-chip bus architecture. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1 –6, 2009. doi: 10.1109/MILCOM.2009.5379966.
27. Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou. Designing and implementing malicious hardware. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–8, Berkeley, CA, USA, 2008. USENIX Association. URL `http://portal.acm.org/citation.cfm?id=1387709.1387714`.
28. F. Koushanfar and A. Mirhoseini. A Unified Framework for Multimodal Submodular Integrated Circuits Trojan Detection. 6(1):162–174, 2011. doi: 10.1109/TIFS.2010.2096811.
29. Farinaz Koushanfar, Azalia Mirhoseini, and Yousra Alkabani. A Unified Submodular Framework for Multimodal IC Trojan Detection. In Rainer Böhme, Philip Fong, and Reihaneh Safavi-Naini, editors, *Information Hiding*, volume 6387 of *Lecture Notes in Computer Science*, pages 17–32. Springer Berlin / Heidelberg, 2010. URL `http://dx.doi.org/10.1007/978-3-642-16435-4_2`.
30. C. Lamech, R. Rad, M. Tehrani, and J. Plusquellic. An Experimental Analysis of Power and Delay Signal-to-Noise Requirements for Detecting Trojans and Methods for Achieving the Required Detection Sensitivities. (99), 2011. doi: 10.1109/TIFS.2011.2136339. Early Access.
31. Jie Li and J. Lach. At-speed delay characterization for IC authentication and Trojan Horse detection. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 8 –14, 2008. doi: 10.1109/HST.2008.4559038.
32. Lang Lin, W. Burleson, and C. Paar. MOLES: Malicious off-chip leakage enabled by side-channels. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 117 –122, 2009a. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5361303`.
33. Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 382–395. Springer Berlin / Heidelberg, 2009b. URL `http://dx.doi.org/10.1007/978-3-642-04138-9_27`.
34. D. Mclntyre, F. Wolff, C. Papachristou, S. Bhunia, and D. Weyer. Dynamic evaluation of hardware trust. In *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, pages 108 –111, 2009. doi: 10.1109/HST.2009.5224990.
35. S. Narasimhan, Dongdong Du, R.S. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia. Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 13 –18, 2010. doi: 10.1109/HST.2010.5513122.
36. Michael Nelson, Ani Nahapetian, Farinaz Koushanfar, and Miodrag Potkonjak. SVD-Based Ghost Circuitry Detection. In Stefan Katzenbeisser and Ahmad-Reza Sadeghi, editors, *Information Hiding*, volume 5806 of *Lecture Notes in Computer Science*, pages 221–234. Springer Berlin / Heidelberg, 2009. URL `http://dx.doi.org/10.1007/978-3-642-04431-1_16`. 10.1007/978-3-642-04431-1_16.
37. Miodrag Potkonjak, Ani Nahapetian, Michael Nelson, and Tammara Massey. Hardware Trojan horse detection using gate-level characterization. In *DAC '09: Proceedings of the 46th Annual Design Automation Conference*, pages 688–693, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-497-3. doi: http://doi.acm.org/10.1145/1629911.1630091.
38. R. Rad, J. Plusquellic, and M. Tehranipoor. Sensitivity analysis to hardware Trojans using power supply transient signals. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 3 –7, jun. 2008a. doi: 10.1109/HST.2008.4559037.
39. R. Rad, J. Plusquellic, and M. Tehranipoor. A Sensitivity Analysis of Power Signal Methods for Detecting Hardware Trojans Under Real Process and Environmental Conditions. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(12):1735 –1744, 2010. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2029117.
40. R.M. Rad, Xiaoxiao Wang, M. Tehranipoor, and J. Plusquellic. Power supply signal calibration techniques for improving detection resolution to hardware Trojans. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 632 –639, 11 2008b. doi: 10.1109/ICCAD.2008.4681643.

41. J. A. Roy, F. Koushanfar, and I. L. Markov. Extended abstract: Circuit CAD tools as a security threat. In *Proc. IEEE Int. Workshop Hardware-Oriented Security and Trust HOST 2008*, pages 65–66, 2008. doi: 10.1109/HST.2008.4559052.
42. H. Salmani, M. Tehranipoor, and J. Plusquellic. New design strategy for improving hardware Trojan detection and reducing Trojan activation time. In *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, pages 66 –73, 2009. doi: 10.1109/HST.2009.5224968.
43. H. Salmani, M. Tehranipoor, and J. Plusquellic. A layout-aware approach for improving localized switching to detect hardware Trojans in integrated circuits. In *Proc. IEEE Int Information Forensics and Security (WIFS) Workshop*, pages 1–6, 2010. doi: 10.1109/WIFS.2010.5711438.
44. H. Salmani, M. Tehranipoor, and J. Plusquellic. A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. (99), 2011. doi: 10.1109/TVLSI.2010.2093547. Early Access.
45. Adam Waksman and Simha Sethumadhavan. Tamper Evident Microprocessors. In *SP '10 Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 173 –188, may. 2010. doi: 10.1109/SP.2010.19.
46. Adam Waksman and Simha Sethumadhavan. Silencing Hardware Backdoors. In *Proc. IEEE Symp. Security and Privacy (SP)*, pages 49–63, 2011. doi: 10.1109/SP.2011.27. URL `http://www.cs.columbia.edu/~simha/preprint_oakland11.pdf`.
47. Xiaoxiao Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic. Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis. In *Defect and Fault Tolerance of VLSI Systems, 2008. DFTVS '08. IEEE International Symposium on*, pages 87 –95, 2008. doi: 10.1109/DFT.2008.61.
48. Sheng Wei and M. Potkonjak. Scalable segmentation-based malicious circuitry detection and diagnosis. In *Proc. MayAugust*, pages 483–486, 2010. doi: 10.1109/ICCAD.2010.5653770.
49. Sheng Wei, Saro Meguerdichian, and Miodrag Potkonjak. Gate-level characterization: Foundations and hardware security applications. In *Proc. 47th ACM/IEEE Design Automation Conf. (DAC)*, pages 222–227, 2010. URL `http://ieeexplore.ieee.org/ielx5/5510861/5522347/05522644.pdf?tp=&arnumber=5522644&isnumber=5522347`.
50. F. Wolff, C. Papachristou, S. Bhunia, and R.S. Chakraborty. Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1362 –1365, mar. 2008. doi: 10.1109/DATE.2008.4484928.
51. Xuehui Zhang and Mohammad Tehranipoor. RON: An on-chip ring oscillator network for hardware Trojan detection. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pages 1–6, 2011. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5763260`.

Integrated Honeypot based Malware Collection and Analysis

Martin Brunner, Christian M. Fuchs, Sascha Todt

Fraunhofer AISEC,
Parkring 4, 85748 Garching (near Munich), Germany
{firstname.lastname}@aisec.fraunhofer.de, sascha.todt@gmx.de
http://www.aisec.fraunhofer.de

Abstract. Timely intelligence on emerging trends in the malicious landscape is an essential prerequisite for successful malware defense and IT early warning. This is commonly gained by collection and examination of current real-world attack data and a preferably meticulous analysis of the most recent samples. However, the ongoing sophistication of malware led to intensive obfuscation and anti-debugging measures and also resulted in a complex and multi-staged malware execution life-cycle. To address this issue we present an ongoing research activity named AWESOME (Automated Web Emulation for Secure Operation of a Malware-Analysis Environment). It is a novel approach for integrated honeypot based malware collection and analysis, intended to cover the entire malware execution life-cycle. Our assumption is that the ability to track this entire life-cycle facilitates a better understanding of current and emerging malware. We introduce our design thereby outlining its benefits as well as design considerations.

1 A Survey on Current Malware

1.1 Initial Situation

Cyber crime has become one of the most disruptive threats today's Internet community is facing. The major amount of these contemporary Internet-based attacks is thereby attributed to malware, which is usually organized within a botnet in large-scale scenarios. Such botnet-connected malware is on their part utilized for infecting hosts and instrumenting them for various malicious activities: most prominent examples are Distributed Denial of Service (DDoS) attacks, identity theft, espionage and Spam delivery [4,16,25]. Botnet-connected malware can therefore still be considered the major threat on today's Internet. Due to the ongoing spread of IP-enabled networks to other areas it can be expected, that the threat posed by botnet-connected malware will intensify and moreover reach further domains in public and private life. Thus, there is a fundamental need to track the rapid evolution of these pervasive malware based threats. Especially timely intelligence on emerging, novel threats is essential for successful malware defense and IT early warning. This requires both, acquisition and examination, of current real-world malware samples in sufficient quantity and variety.

1.2 Malware Evolution

Due to the predominant economic motivation for malicious activities backed by organized cyber crime also the sophistication of malware and the respective propagation methods continuously evolved, hence increasingly impeding malware defense. Thereby the invested effort and the achieved result must be in a reasonable relation for a professional attacker. Thus we experience the phenomena of a moving target. That is, cyber criminals chose their targets and attack vectors according to the best economic relation and an ongoing paradigm shift towards client-side and targeted attacks has been witnessed in recent years [4]. In any way widely spread malware is most effectively managed within a botnet, therefore a newly compromised host is still likely to become a botnet-member. Several work indicates thereby an ongoing specialization of the various groups in the underground economy offering "Malware as a Service" and "pay per install" schemes including elaborated models for pricing, licensing, hosting and rental [5,14,16,17]. This involves professional maintenance, support and service level agreements for the purchasable malware itself as well as innovations in the maintenance of infected victim hosts. Therefore there is (i) one group specializing on the development of the actual malware, (ii) a second group deals with the operation platform and the distribution of malware (i.e., to establish botnets) and (iii) a third group focuses on suitable business models. As a result also the actual malware itself evolved with respect to obfuscation techniques and anti-debugging measures. That is, current malware checks for several conditions before executing its malicious tasks, such as hardware resources of the victim host, Internet connectivity or whether it is executed within a virtualized environment [13,21,29]. In the end this evolution led to a complex and multi-staged malware execution life-cycle.

1.3 Execution Life-Cycle of Modern Malware

With respect to recent advances in malware evolution and findings of related work [5,23] we model the execution life-cycle of today's (autonomous spreading) malware as depicted in Figure 1. A common setting consists of three phases:

(i) *Propagation and exploitation:* This initial phase covers the spread of a malicious payload (e.g., via a worm) that exploits one or multiple vulnerabilities. In this context a vulnerability encompasses technical flaws in operating systems, network services and user applications as well as social engineering techniques. Successful exploitation commonly results in a shellcode getting placed on the victim host which gets then extracted and executed, including possible decryption and de-obfuscation routines.

(ii) *Infection and installation:* As a result of executing the injected shellcode a binary is downloaded by the victim host. This binary is typically a so-called dropper or downloader, which contains multiple malware components and is intended to

disable the security measures on the victim host, to hide the malware components and to obfuscate its activities before launching the actual malware. As there is an emerging trend that multiple cyber criminals instrument a single victim host for their malicious purposes several droppers may be installed (in parallel) within this step. Once the dropper is executed it extracts and installs further components responsible for hardening and updating tasks, thereby preparing the system for the actual malware. After finishing all operations the dropper contacts a remote site in order to validate the victim host and retrieve information on how to retrieve the actual malware. Once downloaded, the malware is executed by the dropper component installing the malware's core components. Finally these core components remove all other (non-vital) components resulting from previous stages and the malware is operational.

(iii) *Operation and maintenance:* Initially, the malware's core components harvest valuable (i.e., marketable) information and send it to a remote server under the control of the attacker in case the attacker loses control over the compromised host later on. Next, the malware attempts to establish a C&C channel awaiting further instructions, such as launching malicious actions and maintenance operations.

The various steps of the outlined malware execution life-cycle include many checks and measures each intended to maximize the success of the malware installation, ensuring a reliable operation and to protect the cyber criminals from being tracked down. In particular there are several apparent advantages for an adversary. First, there is no need to distribute the core malware components in the first phase thereby impeding successful collection and thus detection and mitigation of the malware. In addition the malware can be distributed more selectively and targeted thereby ensuring the victim host's authenticity. After all, the outlined malware execution life-cycle introduces several challenges for malware collection and analysis.

2 Challenges in Malware Capture and Analysis

Honeypots have been proven to be a fundamental part for malware collection as they can provide valuable information on current attacks that would have been difficult to acquire otherwise [28]. This enables further analysis and examination of the collected malware samples finally resulting in the ability to anticipate recent trends in the malicious landscape. While mostly server based, low-interaction (LI) honeypots have been instrumented for the collection of undirected, widely spread malware, they are limited in the scope of attacks they can cover. Hence new types of honeypots, such as client honeypots and high-interaction (HI) honeypots, emerged to face the evolving sophistication of malware. Especially the latter can provide more valuable information than LI honeypots while posing much more risk for the operator as well as for third party systems getting compromised.

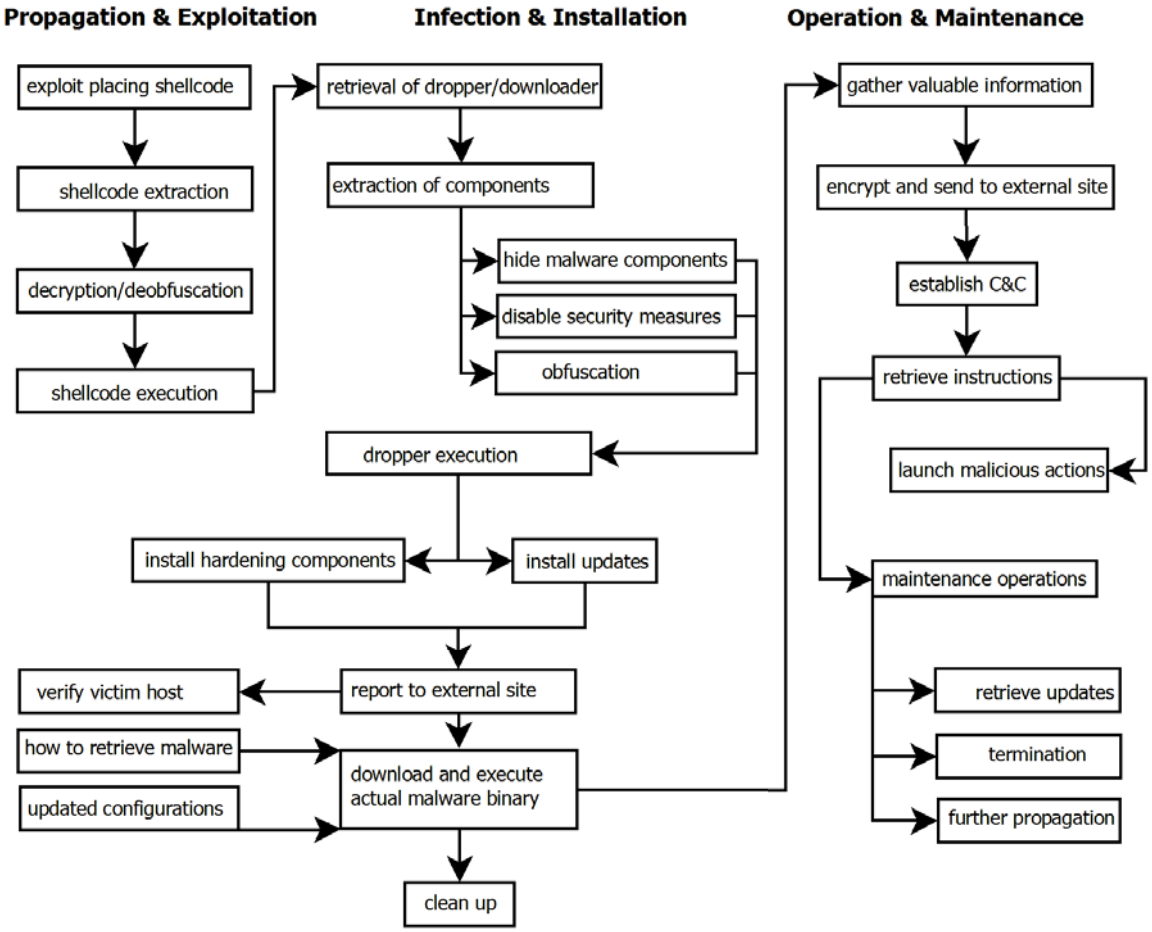


Fig. 1. Execution Life-Cycle of Modern Malware

A major issue, that makes malware analysis a challenging task, is the ongoing arms race between malware authors on the one hand and malware analysts on the other hand. That is, while analysts use various techniques to quickly understand the threat and intention of malware, malware authors invest considerable effort to camouflage their malicious activity and impede a successful analysis [8,21].

In addition malware should ideally be granted unhindered access to all requested resources during runtime in order to gain a comprehensive analysis covering the full life-cycle. While this could easily be achieved by allowing full interaction between the malware and Internet-connected third party systems, this is not a viable approach in setups which can not neglect liability issues. However, if a given malware sample can not retrieve all requested resources during its execution life-cycle, it may behave different or refuse to execute at all.

Infrastructures for large-scale malware collection can satisfy the requirements for automated tracking of malware, which has been demonstrated by existing, ad-

vanced approaches [1,7,12] handling vast amounts of malware of varying sophistication. Complementary to the therein covered research topics our presented approach addresses the following issues:

(i) Despite being executed within an isolated environment a given sample must be supplied with (all) requested network services during analysis. Otherwise achieving high quality results may be impeded due to a restriction of resources and lead to different malware behavior or even a refusal of execution. While certain services can be offered using sinkholing techniques, such approaches are usually purely network based. Thus reactions to malware initiated attempts remain static during runtime. That is, they present pre-defined, commonly used services which are usually queried, and are unable to handle any other requests accordingly.

(ii) HI honeypots pose, beside complexity and maintenance issues, a high operational risk which is often inadequately addressed. While there are several ways to mitigate this issue, the remaining risk is still higher compared to LI honeypots. Beside ethical aspects, this also causes legal and liability issues for the operating organization.

(iii) Malware collection and analysis is commonly separated, thus losing the system context (i.e., file handles, requests, sockets) of the victim host. While such separation is not necessarily a limitation (i.e., may not be mandatory to gain qualitative analysis results), we argue this loss of information hinders analysis, may degrade analysis results or even prevent analysis of certain malware.

3 A Holistic Approach for Integrated Malware Collection and Analysis

3.1 Goals

Our overall goal is to capture and dynamically analyze malware at a large-scale in order to identify trends of current and emerging malware. Thereby we aim to cover the entire execution life-cycle of novel malware in an automated way within a controlled environment. That is, we want to track malware communicating via unknown (C&C) protocols as well. In order to minimize harm to third parties malware should by default have no Internet access during analysis. The whole procedure intends to trick a sample into believing to run on a real victim host with full Internet access.

3.2 Basic Concept

In order to identify the services and protocols required in the next step within the execution life-cycle, we intend to harvest information on malware logics directly during execution. Contrary to purely network-based concepts, our approach additionally operates at binary level, directly interacting with the malware's host system.

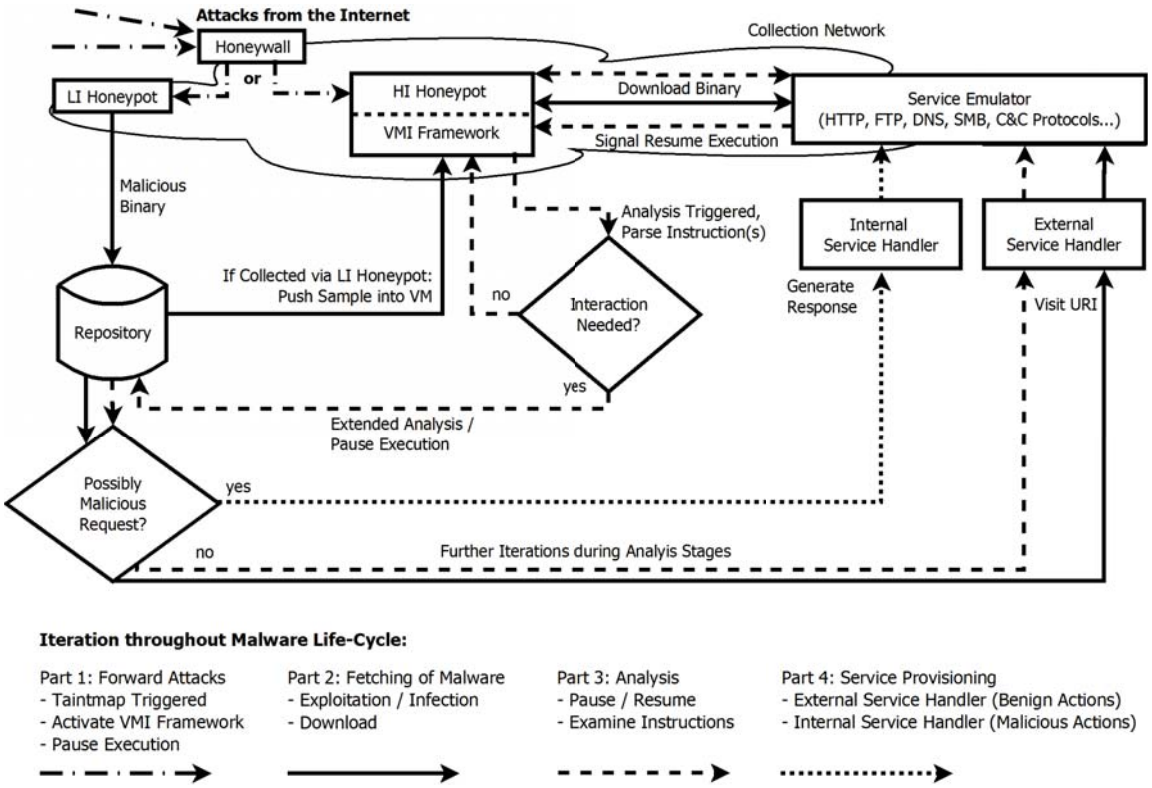


Fig. 2. General Design of the Presented Approach

Thus it aims to integrate network-based analysis and binary analysis, as in [30]. As depicted in figure 2, the key component of AWESOME is based on a HI honeypot and a virtual machine introspection (VMI) framework. It integrates the honeypot based malware collection with dynamic malware analysis. This system is designed to enable automated malware collection and analysis, preserving the context of the exploited system. We enhance the solution with a transparent pause/resume functionality which is instrumented to determine and - if appropriate - interrupt the program flow. This enables extraction and alteration of program logics and data within the victim environment during runtime. It is specifically valuable for extracting protocol information and cryptographic material used by the malware, in order to determine the used protocol type and to intercept encrypted communication. Extracted protocol information is forwarded to a service handler (SH) and sinkholing service in order to maintain full control over all interactions between the malware and the outside world. For handling unknown traffic as well, finite state machines (FSM) are automatically derived from the observed traffic and used for service emulation. Automating the whole process of integrated collection and analysis aims towards handling large amounts of malware, thus making our system scalable.

3.3 Added Value

The system context of the malware collection facility persists, and is also used in the subsequent analysis. Integrated collection and analysis is similar to the approach used in HI honeypots and thus is more closely aligned to real-world scenarios. In addition we achieve (increased) transparency during analysis due to the use of VMI. We consider this to be a benefit, since we argue that VMI based analysis is more likely to remain undetected by malware and requires no trusted supportive components inside the sample's context of execution, compared to other techniques, that have been evaded [11]. Hence the approach is more likely to observe the entire malware execution life-cycle. Furthermore we are able to extract and inject data as well as instructions from or into the memory of the infected virtual machine (VM) during runtime, which can be used to tap and manipulate encrypted C&C traffic. Since our approach depends on no analysis components within the VM we believe it to be more secure, while also expecting better overall performance. Moreover we are able to control any interaction between the malware and third party systems and thus fulfill legal and liability constraints. Since our approach is applied directly at the instruction level we are aware of the actions initiated by the malware. Thus we can provide the according services and even service novel communication patterns. After all, the risk resulting from HI honeypot operation is minimized.

3.4 Design and Implementation

Setup For *malware collection* we chose to apply the taint-map based approach of the ARGOS HI honeypot [26] to achieve our primary goal, i.e., handling unknown malware. While popular LI honeypots have proven to be efficient means for malware collection, their knowledge-based approach has also drawbacks regarding the quantity and diversity of the collected malware [32]. ARGOS allows the detection of both known and unknown (0-day) attacks but is independent of special collection mechanisms. *Malware analysis* is conducted based upon Nitro [24], a KVM-based framework for tracing system calls via VMI. In particular we determine whether a given action initiated by the currently analyzed malware requires Internet access. Since Nitro is based on KVM, it can cooperate with ARGOS, which relies on QEMU. The *service provisioning* component manages all malware initiated attempts requesting resources on the Internet. Malicious attempts are handled via an appropriate sinkholing service spawned by Honeyd [27] and unknown traffic patterns may be handled utilizing ScriptGen [20]. For our setup we made several modifications to the utilized components.

(i) Since ARGOS is more time-consuming than traditional approaches and thus detectable by an abnormal latency and timing-behavior, a pause and resume function has been implemented.

(ii) Once the taint-map reports tainted memory being executed, we activate the analysis functionality, provided by the VMI framework.

(iii) Simple interpretation and filtering of system calls and their parameters is conducted directly within hypervisor space, while more complex analysis is performed via the VMM in the host environment [15].

The entire process consists of three parts (collection, analysis and service provisioning) and is structured as described below. The steps are repeated, thereby iterating throughout the entire life-cycle of the malware.

Malware Collection To overcome the poor performance of ARGOS we deploy a two staged malware collection network, i.e., a hybrid honeypot system, similar to existing approaches such as [2,18,31]. We take advantage of our preexisting honeyfarm infrastructure [3] which utilizes a large-scale network telescope employing various different LI honeypots. This infrastructure is used to filter noise and known, thus uninteresting attacks. Only novel incidents are forwarded to ARGOS, thus reducing the overall load on it.

Malware Analysis Dynamic malware analysis utilizing virtualization can be recognized and thus evaded by environment sensitive malware [13,15,21,29]. Hence our goal is to achieve a preferably transparent dynamic malware analysis, whereas we consider VMI as the currently most promising approach to evade malware's anti-debugging measures. We chose Nitro since it offers several advantages regarding performance and functionality, compared to other publicly available tools such as Ether [10], as stated in [24]. As Nitro is based on KVM we have - beside guest portability - full virtualization capability thanks to the host CPU's virtualization extensions and expect reasonable performance. During the analysis process we expect a malicious binary to be shellcode or a dropper rather than the actual malware binary. This initially retrieved binary is then decoded and usually contains a URL pointing at the resource used for deploying the next stage of the malware. In the second iteration, execution of this binary continues after it has been downloaded and the VM has been resumed. The resulting system call trace produced by Nitro is then examined for routines related to connection handling. If present we transparently pause execution of the VM and forward related traffic to the service provisioning component.

Service Provisioning Malware-driven outbound requests are evaluated to prevent harm to third party systems. For these checks we rely upon existing measures, such as IDS or a web application firewall. As we are well-aware that such measures won't be sufficient to tell apart benign and malicious flows in every case, we may build on existing approaches, such as [19]. We assume that a purely passive request (e.g., a download) on a throttled Internet uplink does not cause any harm to a third party. It is thus considered to be benign and handed over to the *external service handler* (see figure 2). Since the external SH has Internet access, it resides in a dedicated

network segment separated from the analysis environment. If a given request can not be determined to be benign it is redirected to the *internal service handler*. The sole task of these SHs is to fetch, prepare or provide information for the *service emulator* (SE). The SE launches the requested service in order to deliver the appropriate payload supplied by the SH. Afterwards the execution is transparently resumed. Since these services can be extremely heterogeneous the SE is based on Honeyd as a very flexible and scalable tool which is able to emulate or spawn arbitrary services, given that a protocol template exists. The creation of templates for novel protocols is a much more challenging task. Therefore we plan to instrument a tool, which derives FSMs off observed traffic, such as *ScriptGen* or a similar approach [6,9,22]. Thereby, each FSM represents the behavior of a given protocol on an abstract level while not depending on prior knowledge or protocol semantics. Based on the generated FSMs service emulation scripts for use in the SE can be derived. By integrating such a tool into our approach we aim towards adding 'self-learning capabilities' to the service provisioning element. Obviously this (at least) requires one-time observation of a given communication between the honeypot and the external system. Hence a supervised back-channel for learning about novel protocols is needed. Once this has been established and the appropriate FSM has been generated, we are able to handle the new protocol as well. While this is a clear limitation we consider it to be a reasonable trade-off.

4 Summary

We presented a novel approach for integrated honeypot based malware collection and analysis which addresses several issues, namely the separation of collection and analysis, the limitations of service emulation and the operational risk of HI honeypots. The overall goal is to capture and dynamically analyze malware at a large-scale while covering the entire execution life-cycle of a given malware. The key contribution of the approach is the design of the framework as well as the integration and extension of the stated tools. While this is an ongoing research activity and thus still under development, several modifications to ARGOS and Nitro have already been implemented and successfully tested indicating the feasibility of our approach. Future work will include the completion and evaluation of the service emulation component and the measures to prevent harm to third party systems. In particular we will evaluate all described components and develop measures to validate the rule set for analyzing outgoing requests.

References

1. M. Apel, J. Biskup, U. Flegel, and M. Meier. Early warning system on a national level - project amsel. In *Proceedings of the European Workshop on Internet Early Warning and Network Intelligence (EWNI 2010)*, January 2010.

2. M. Bailey, E. Cooke, D. Watson, F. Jahanian, and N. Provos. A hybrid honeypot architecture for scalable network monitoring. 2006.
3. M. Brunner, M. Epah, H. Hofinger, C. Roblee, P. Schoo, and S. Todt. The fraunhofer aisec malware analysis laboratory - establishing a secured, honeynet-based cyber threat analysis and research environment. Technical report, Fraunhofer AISEC, September 2010.
4. BSI. Die lage der it-sicherheit in deutschland 2011. Bundesamt fuer Sicherheit in der Informationstechnik, May 2011.
5. J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: the commoditization of malware distribution. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, Berkeley, CA, USA, 2011. USENIX Association.
6. J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 621–634, New York, NY, USA, 2009. ACM.
7. D. Cavalca and E. Goldoni. Hive: an open infrastructure for malware collection and analysis. In *proceedings of the 1st workshop on open source software for computer and network forensics*, 2008.
8. X. Chen, J. Andersen, Z. Mao, M. Bailey, and J. Nazario. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 177 –186, june 2008.
9. W. Cui, V. Paxson, N. C. Weaver, and Y. H. Katz. Protocol-independent adaptive replay of application dialog. In *In The 13th Annual Network and Distributed System Security Symposium (NDSS*, 2006.
10. A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62, New York, NY, USA, 2008. ACM.
11. M. Dornseif, T. Holz, and C. Klein. Nosebreak - attacking honeynets. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, june 2004.
12. M. Engelberth, F. Freiling, J. Göbel, C. Gorecki, T. Holz, R. Hund, P. Trinius, and C. Willems. The inmas approach. In *1st European Workshop on Internet Early Warning and Network Intelligence (EWNI)*, 2010.
13. P. Ferrie. Attacks on virtual machine emulators. In *AVAR Conference, Auckland*. Symantec Advanced Threat Research, December 2006.
14. J. Franklin, A. Perrig, V. Paxson, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, New York, NY, USA, 2007. ACM.
15. C. M. Fuchs. Deployment of binary level protocol identification for malware analysis and collection environments. Bacherlor's thesis, Upper Austria University of Applied Sciences Hagenberg, May 2011.
16. P. Gutmann. The commercial malware industry. In *DEFCON 15*, 2007.
17. T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. Technical report, University of Mannheim, Laboratory for Dependable System, 2008.
18. X. Jiang and D. Xu. Collapsar: a vm-based architecture for network attack detention center. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, Berkeley, CA, USA, 2004. USENIX Association.
19. C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. Gq: practical containment for measuring modern malware systems. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, New York, NY, USA, 2011. ACM.
20. C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, Washington, DC, USA, 2005. IEEE.
21. M. Lindorfer, C. Kolbitsch, and P. Milani Comparetti. Detecting environment-sensitive malware. In *Recent Advances in Intrusion Detection (RAID) Symposium*, 2011.
22. P. Milani Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol specification extraction. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pages 110–125, Washington, DC, USA, 2009.

23. G. Ollmann. Behind today's crimeware installation lifecycle: How advanced malware morphs to remain stealthy and persistent. Whitepaper, Damballa, 2011.
24. J. Pfoh, C. Schneider, and C. Eckert. Nitro: Hardware-based system call tracing for virtual machines. In *Advances in Information and Computer Security*, volume 7038 of *Lecture Notes in Computer Science*. Springer, Nov. 2011.
25. D. Plohmann, E. Gerhards-Padilla, and F. Leder. Botnets: Detection, measurement, disinfection & defence. European Network and Information Security Agency (ENISA), 2011.
26. G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. In *Proc. ACM SIGOPS EUROSYS'2006*, Leuven, Belgium, April 2006.
27. N. Provos. A virtual honeypot framework. In *Proceedings of the 13th USENIX Security Symposium*, 2004.
28. N. Provos and T. Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley, 2008. ISBN 9780321336323.
29. J. Rutkowska. Red pill... or how to detect vmmusing (almost) one cpu instruction, 2004. http://invisiblethings.org.
30. D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. Bitblaze: A new approach to computer security via binary analysis. In *Proceedings of the 4th International Conference on Information Systems Security.*, 2008.
31. M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *Proceedings of the 20th ACM symposium on Operating systems principles*, SOSP '05, New York, NY, USA, 2005. ACM.
32. J. Zhuge, T. Holz, X. Han, C. Song, and W. Zou. Collecting autonomous spreading malware using high-interaction honeypots. In *Proceedings of the 9th international conference on Information and communications security*, ICICS'07, Berlin, Heidelberg, 2007. Springer-Verlag.

Predentifier: Detecting Botnet C&C Domains From Passive DNS Data

Tilman Frosch, Marc Kührer, and Thorsten Holz

Horst Görtz Institute (HGI),
Ruhr-University Bochum, Germany
{firstname.lastname}@rub.de

Abstract. The Domain Name System (DNS) is mainly used for benign and legitimate Internet activities. Nevertheless, it also facilitates malicious intentions. Domain names have started to play an increasingly important role in the Command and Control (C&C) infrastructure of botnets. These domains can be added to blocklists or taken down, yet attackers can simply evade the countermeasures by creating hundreds of new domains every day.
We propose a framework called Predentifier to detect C&C domains at an early stage. It combines a host's DNS configuration properties with secondary data to derive a set of distinctive features that can be used to describe the behavior of a host. We employ methods of statistical learning to determine with high reliability, whether a domain belongs to a C&C server or if it is benign. We further show that it is possible to leverage passive DNS data to identify C&C domains without infringing on employment or customer rights.

1 Introduction

In recent years, domain names have started to play an increasingly important role in *Command and Control* (C&C) mechanisms of *botnets*, i.e., networks of compromised machines under the control of an attacker (often called *botmaster*) [6,8,15]. Botnets are responsible for some of the major problems on the Internet: they are used to propagate spam and to steal of banking credentials and accounts to a variety of online services, among other criminal activities like *Distributed Denial of Service* (DDoS) attacks [14]. Many botmasters today maintain control over their criminal assets by using DNS-based C&C structures to prevent efforts to take down the botnet. A timely identification of C&C domains can allow for the detection of a botnet even before it is put to use on a large scale. We introduce Predentifier to effectively identify C&C domains at an early stage – without infringing on employment or customer rights. The approach combines DNS configuration properties of a host with secondary data like WHOIS and geolocation information. We derive a set of 14 distinctive features and employ methods of statistical learning to decide with a high confidence, whether a domain is used for C&C or is affiliated with a benign, legitimate participant on the Internet.

2 Related Work

The idea of performing passive DNS replication to detect malware was introduced by Weimer [16] in 2005. Zdrnja et al. [19] adopted this idea and proposed a passive

DNS system to trace hosts associated to botnet C&C servers. To identify hosts and domains which are participating in Fast-Flux networks, Holz et al. [11] analyzed DNS records aggregated with further information like *Autonomous System Numbers* (ASN) and geolocation data. Yadav et al. [18] proposed an approach to detect domain fluxes in DNS traffic by identifying domain names which have been generated algorithmically. Felegyhazi et al. [7] investigated DNS properties and registration information to explore the potential of proactive domain blacklisting. Antonakakis et al. [3] presented the dynamic reputation system NOTOS based on DNS and secondary data provided by honeynets and malware analysis services, which analyzes network and zone features to describe characteristics of domains. In a second paper, Antonakakis et al. [4] introduced a system which attempts to detect malicious domains by analyzing passive DNS data gathered at authoritative nameservers and top-level domain servers. Bilge et al. [5] proposed the architecture EXPOSURE which analyzes passive DNS data to automatically distinguish between malicious and benign domains. The authors introduce time-based and DNS-based features.

Besides implementations using (passive) DNS data, other approaches have been published focusing on lexical and host-based features to distinguish between benign and malicious hosts [12,10].

3 Motivation: DNS Features of Botnet Domains

The intuition of our approach is that the requirements towards hosts used for controlling botnets differ from the requirements a user has towards a server providing benign content. These requirements are also reflected in the DNS configuration. While a well-established site rarely changes the IP address(es) it resolves to, the maintainer of a botnet C&C server may suddenly need to change an `A`-record in order to maintain continuous control over the bots. Be it because the legitimate owner of a compromised host used as C&C server disconnects his system from the Internet, or that a server, legitimately acquired by the botherder, is seized by law enforcement officials in an attempt to shut down the botnet. As such a configuration change needs to propagate quickly, this policy is reflected in a lower TTL value in the DNS configuration and may, for example, also be reflected in the *refresh* value that determines the time between two zone transfers requested from secondary nameservers.

Redundancy is defined differently for benign and C&C domains. This is reflected in the amount of IP addresses a domain resolves to. A benign site can balance load by simply resolving a hostname to a set of IP addresses in a round-robin fashion. High-traffic sites are also often hosted on *Content Delivery Networks* (CDN) and behave differently compared to ordinary hosts offering benign content [11]. Botnet C&C servers will receive fewer requests than high-traffic sites, thus do not need load-

balancing in the same way. What the botherder needs instead is a way to mitigate takedowns of C&C servers currently in use. This can be achieved by changing the `A`-record for this hostname. As a consequence, the average amount of IP addresses being resolved for one malicious hostname at a time may be lower than the amount of addresses seen in the context of a benign domain. However, this assumption highly depends on the data set and whether there are any active *fast flux* [17] domains in the set of malicious domains.

A hypothesis in the context of WHOIS data implies that benign domains tend to be older than domains used for botnet C&C purposes [5]. While analyzing the registration dates of the domains in our data sets, we indeed found that on average the age of C&C domains is significantly lower.

Other properties that are expected to differ for benign and malicious domains are the geographic locations of the IP addresses being resolved from one hostname and their locations within the network as reflected by the ASNs. Servers used for benign purposes and addressed by the same domain are often located in the same country. This still holds true for massively loadbalanced setups and CDNs. Further on, a legitimate business might rather choose to have all its hosting services provided by one organization which will result in only a few ASNs or even only one. In contrast, a botmaster tends to act opportunistic and use any host that is available and fits his needs. A widely distributed architecture does not proof disadvantageous – on the contrary, spreading C&C operations over more than one legislation may increase the resilience against takedowns by law enforcement.

4 System Design

To generate detection models for C&C domains based on the assumptions and observations in the previous section, Predentifier applies 14 distinctive features derived from passive DNS, WHOIS, and geolocation data.

4.1 System Outline

Figure 1 outlines the architecture of Predentifier. In the offline phase, one of the core components of our analysis system is the database storing labeled benign and malicious domains. We generate a training set $\mathcal{S}_{\mathcal{TRAIN}}$ by randomly choosing labeled benign and malicious domains from the database. We then attribute features to the individual hostnames, based on passive DNS, WHOIS, and geolocation information. From these information we create a detection model to be used by the classifier. In a next step, we choose a test set $\mathcal{S}_{\mathcal{TEST}}$ of domains and also attribute features to each element of the set. However, the labels $Y_i^{'}$ are removed from the set. We classify all elements of $\mathcal{S}_{\mathcal{TEST}}$ and compare the inferred class Y_i with the original labels $Y_i^{'}$ of the test set to determine true and false positives. When used

as an online system, several passive DNS sensors, deployed in various networks, collect DNS answers from the network traffic. Again, features are attributed to these hostnames, which are then classified based on the previously built model.

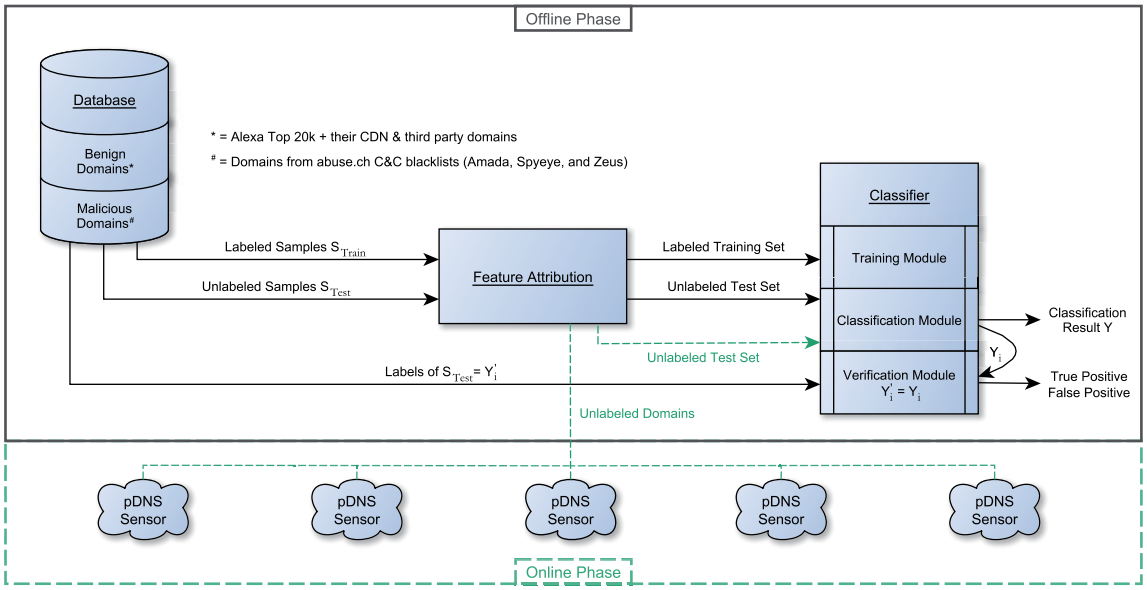


Fig. 1. System architecture of PREDENTIFIER

4.2 Data Acquisition

A very common approach to acquire a set of benign hostnames is to use the TOP 20.000 domains from the Alexa traffic ranking [2]. Most of these sites, however, make use of CDNs and other sites providing third-party content that consequently are equally highly frequented as the site they serve content to. Some of these may exhibit similar behavior than malicious domains, e.g., in terms of zone configuration. We include the *Fully Qualified Domain Name* (FQDN) of each host that provided content to these sites, so the training and test sets also contain these domains and the model is created accordingly. Additionally to the 20.000 primary domains, we found 40,881 domains that are used to provide third-party content to the high-traffic domains or were part of a CDN. We verified that none of the resulting 60,881 domains was found on any of the publicly available C&C blacklists.

As *malicious* domain set we use a subset of domains accumulated from three C&C server blacklists provided by abuse.ch [1]: their Malware Database C&C blocklist (AMaDa) as well as the dedicated ZeuS and SpyEye trackers.

4.3 Classifier

For the classification process we use *k-Nearest Neighbor* (kNN). kNN is one of the most straight-forward supervised learning methods. The simplified basic assumption is that samples defined by an n-dimensional vector which are closest in this vector space must be similar, i.e., it determines the decision boundary locally. The parameter k describes how many neighboring points should be taken into account. Equation 1 shows the k-nearest neighbor fit Y for a classification for an unclassified sample x, where $N_k(x)$ is the neighborhood of x defined by the k-closest points in the training set [9].

$$Y(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{1}$$

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

A common metric to define closeness is the Euclidean distance. Equation 2 shows the Euclidean distance d of two vectors x, y consisting of values x_i, y_i.

A classification with $k = 1$ is not very robust as any new sample x is simply assigned to the class of the nearest element in the training set. kNN with $k > 1$ is more robust as it assigns samples to the majority class of their k-closest neighbors [13]. A low value k will introduce more noise intro the results. On the other hand, a high value of k renders kNN computationally more expensive. It also conflicts the basic idea behind kNN, i.e., that points, which are near to each other, are more likely to belong to a similar class than points with a higher distance.

4.4 Feature Selection

As already outlined in Section 3, properties observed from passive DNS data and other sources reveal information to describe either malignance or benignity of a domain. However, the choice among possible features is strongly determined by data availability and level of disaggregation: the passive DNS data available to us is aggregated with respect to the temporal dimension. Thus, we are not able to analyze patterns of repeated usage, daily similarity in when and how often a domain is accessed and similar features. Instead, we introduce a number of other features that evolved from the available data.

As shown in Table 1, we identified a set of 14 distinct features of which nine features have, to the best of our knowledge, never been tested in previous work and can be considered novel. Fig. 2 outlines the arithmetic average and standard deviation for each feature normalized to the *global average* of each feature, i.e., the arithmetic average of the feature value as found in the combination of both

#	Feature Name	Description
1	digitratio	Number of digits compared to length of domain
2	consonantratio^{novel}	Number of consonants compared to length of domain
3	consonantvocalratio^{novel}	Number of consonants compared to amount of vocals
4	ipcount	Number of IP addresses the domain resolves to
5	ttl_max^{novel}	Maximum TTL during observation (in sec.)
6	ttl_min^{novel}	Minimum TTL during observation (in sec.)
7	ttl_diff^{novel}	$Max.\ TTL - min.\ TTL$ during observation (in sec.)
8	soa_sn_changes^{novel}	Number of SOA S/N changes during observation
9	exp_time_min^{novel}	Minimum expiry time of domain's zone (in sec.)
10	rtry_time_min	Minimum retry time of domain's zone (in sec.)
11	rfrsh_time_min^{novel}	Minimum refresh time of domain's zone (in sec.)
12	countrycount	Number of countries the domain is served from
13	asncount	Number of ASNs the domain is served from
14	domainage^{novel}	*last_seen date* − *domain creation date* (in days)

Table 1. Features utilized by PREDENTIFIER

the benign and the malicious set. Normalization provides a better overview of the differences between malicious and benign domains and eases the interpretation of the different features as the features vary widely in scale.

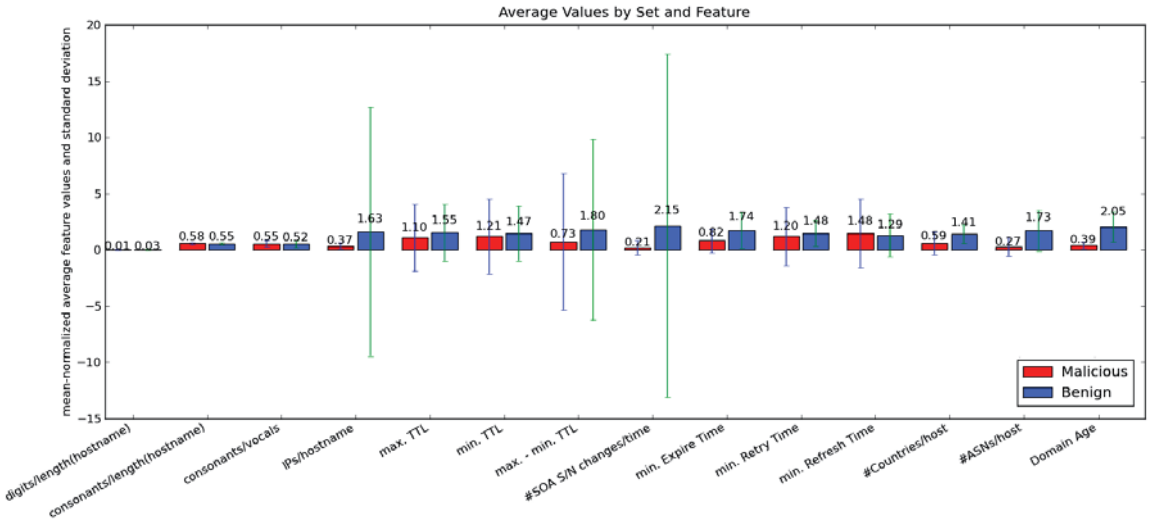


Fig. 2. Mean-normalized average feature values and standard deviation

Lexical Features: Ma et al. [12] justify lexical features based on the observation that malicious URLs tend to look differently compared to benign URLs. Our approach does not deal with URLs as a whole but with a subset of the URL: the domain. The argument is also confident for the domain since it may look different, especially for domains resulting from a *Domain Generation Algorithm* (DGA). Our lexical

features include the number of digits compared to length of the domain, the amount of consonants compared to length, and the number of consonants compared to the amount of vocals, based on the assumption that these ratios differ between words of a spoken language and (randomly) generated strings.

DNS Answer-based Features: Feature 4 consists of the number of IP addresses a domain is resolved to during the observation period. The intuition for using this feature is that legitimate websites receiving significant traffic tend to use several hosts in parallel to balance load. A round-robin resolution appears as `A`-records of one domain resolved to different IP addresses, where each database entry exhibits a *last_seen* timestamp of similar age. This still holds true for CDNs, although only for a given geographic and network-topologic location. The amount of hosts that deliver the same content to a user in a given area via a given ISP varies widely depending on the time of day [11]. However, aggregated over an observation period of seven to 30 days, the amount of individual active IP addresses stays relatively constant for a given website. Consequently, CDN hosts and traditionally loadbalanced sites can be assumed to behave similar with respect to feature 4. The number of IP addresses per domain, however, varies highly depending on the actual domain name. For a botmaster it is crucial that only few of the C&C servers appear in public. Thus, a domain used in a C&C context is more likely to be resolved to only few IP addresses at a time. The distinctiveness of this feature alone is of course limited, as small benign websites also use just one IP address.

IP Address-based Features: IP address-based features are attributed to a domain indirectly, such as the number of different countries and ASs a client is served from. Sample data indicates that the amount of different countries and ASs a domain is affiliated with is higher for benign domains. Yet, botmaster may take advantage of heterogeneous jurisdictions by spreading C&C servers over several countries or using any host available. This observation is also shared by Bilge et al. [5] with regard to geolocation and by Antonakakis et al. [4] with regard to ASs.

Some high-traffic websites might do load-balancing multi-nationally, but most will simply make use of CDNs to reduce load and latency. It follows that again the large majority of benign domains is being served from one country and one AS. However, the passive DNS database is fed by sensors at a variety of locations which may each see a different set of IP addresses associated with a domain. As a consequence, the amount of different ASNs and countries associated with a domain is on average slightly higher for benign domains. However, standard deviation indicates that these features can be volatile, as there exist many benign hostnames pointing to exactly one IP address located in one country.

Zone-based Features: These features are derived from SOA resource records and cover the minimum and maximum TTL, the difference between those values as well

as the number of serial number changes for the respective SOA record, and the minimal values for expiry time, retry time, and refresh time.

A lower TTL value allows for a higher flexibility as it reduces caching time on the client side, e.g., for hostname ↔ IP address relations. For a benign domain, the DNS configuration for an individual host will seldom change. Under these premises, a high TTL reduces the load on the domain name system while the slow propagation of changes is tolerable. In a C&C setting, the maintainer may be confronted with less foreseeable events that make it necessary to update a record faster. The feature, however, is not distinctive on its own: a very low TTL is also used for load-balancing via round-robin DNS [5]. Besides using the maximum and the minimum of a domains's TTL, we record the difference between both TTLs which indicates a change in zone behavior.

Refresh time, retry time, and expiry time are similar indicators for desired flexibility but on a nameserver infrastructure level. The refresh time is the equivalent of TTL for SOA records, i.e., the time a secondary nameserver will wait before querying the primary server for changes. The retry time determines how long a host will wait before retrying after a failed zone transfer. 80% of the domains in the malicious set are configured with a $\textit{retry time} \leq 1800\ \textit{seconds}$ while the same is true for only about 21% of the benign samples. The tendency towards lower retry times observed with malicious zones can again be interpreted as an attempt to make the complete setup more resilient against failures and offer the possibility for faster recovery. The expiry time determines how long a secondary nameserver will continue to try to complete a zone transfer from a primary server. The average expiry time in the benign random set is by factor 2.13 higher than in the malicious set, and the standard deviation of the non-normalized values indicates that the respective value intervals do not overlap.

The amount of SOA serial number changes indicates how often a zone is updated during the observation period. Our data indicates that this differs for benign and malicious zones, thus we use the amount of serial number change as a feature.

WHOIS-based Features: The registration date of a domain is provided by many registries in the WHOIS answer and can be used to calculate the age of a domain. Bilge et al. [5] observe that malicious domains often have a short life span. It follows that one can expect to observe a lower age in this group of domains when compared to benign domain names that are often part of a well-established infrastructure. While we have observed domains blacklisted for C&C with a domain age above 3000 days, the averages of the benign and malicious random set vary in the scale of years. The average age of a benign domain in the sample is 2276 days, while the average age of a malicious domain is 416 days with a median of 306 days (2086 days for the benign set). Also, only 3% of the random benign domains are up to a year old or younger while this is true for more than 62% of the malicious domains.

5 Evaluation

We first test the soundness of the approach by performing 10-fold cross validation on different sets of hosts. We then test the effectiveness in a realistic scenario. In a real-world scenario, PREDENTIFIER would be required to train on data observed in the past, i.e., the time period $[t.now - n, ..., t.now]$ and use the resulting model to correctly identify C&C servers from data observed in the time period $[t.now + 1, ..., t.now + m]$, where $t.now$ describes the time being, n is a value indicating the end of the observation period in the past, and m is the time period after that the classifier is re-trained. We simulate this scenario by using data from domains observed between a date in the past d_p and a date even further in the past $d_p - n$ to compile a training set and test it against data observed from domain names first seen at the time $d_p + 1$ until the day we performed the evaluation. In a next step, we evaluate the contribution our novel features offer to the classification results.

5.1 Evaluation Data

In the following, we refer to the 60,881 domains derived from the TOP 20,000 domains in the Alexa traffic ranking as the *global benign set* $\mathcal{B}$. Furthermore, we refer to the set of samples associated with any entry in any of the abuse.ch blacklists as the *global malicious set* $\mathcal{M}$. We refer to a labeled set of samples as *training set* $\mathcal{S}_{\mathcal{TRAIN}}$. When a set is referred to as *test set* $\mathcal{S}_{\mathcal{TEST}}$ it is an unlabeled set of samples. The classification process assigns a sample $x \in \mathcal{S}_{\mathcal{TEST}}$ either to the benign or the malicious class. We calculate the *false positive rate* as the percentage of benign samples that is falsely classified as malicious and the *detection rate* or *true positive rate* as the percentage of malicious samples that is correctly classified.

5.2 Cross Validation

N-fold cross validation is a technique frequently used to evaluate the effectiveness of a proposed detection mechanism.Applying this technique splits the data into N random partitions. $N - 1$ partitions are used to train the classification algorithm, and the resulting model is then tested on the remaining partition. The N resulting detection and false positive rates are then averaged.

We use 10-fold cross validation to evaluate the detection accuracy on data observed during a 30-day period. In the context of this evaluation, we also determine which value k in kNN yields the best classification results. For every test, the cross validation set is the same and consists of identical partitions for every run, i.e., we randomize and partition the set once and keep this configuration for every run. Thus, the only parameter that varies between the different cross validations is the value of k.

The sets used for the cross validation contained the following amount of samples: $|\mathcal{S}_{\mathcal{TRAIN}}| = 17487$ and $|\mathcal{S}_{\mathcal{TEST}}| = 1943$. These two sets consist of a total of 17,386

benign and 2,044 malicious domains. We evaluated false positive and detection rate for classifications with kNN, where $k \in [1, 2, ..., 15, 50, 150, 1500]$. On this data set a downward trend can be observed for $k > 2$.

The best detection rate of 93.6% in combination with a low false positive rate of 0.5% was achieved with $k = 2$. The results from the 10-fold cross validation on these two different sets show that the approach is indeed fit to detect botnet C&C servers from passive DNS information in combination with secondary data. The test on the second data set shows that good results can be achieved both when classifying hosts, whose activity has been detected temporally near to each other and also when classifying hosts that may last have been observed in a temporal distance of up to 30 days.

5.3 Temporally Disjoint Data Sets

This kind of evaluation can be considered a reality check for our approach. A system based on this approach should be able to successfully detect C&C servers it observes in the future based on training data observed in the past. In order to represent this scenario in a realistic manner, we randomly choose ten training sets, each consisting of 500 benign and 500 malicious samples from domains observed between 2011-12-07 and 2012-01-15. We use the feature values determined for these samples to calculate global average values for each feature. As test set we use all domains observed between 2012-01-16 and 2012-02-07 that fulfill the requirement of having at least five determinable host-based features. From this data we randomly choose ten training sets and ten identical test sets. The latter consist of 1,185 benign and 822 malicious samples.We run kNN using each of the training sets as input and test the resulting model on the test set. The evaluation result is reported as the average detection and false positive rate calculated over these ten runs. In a real-world application, one would also draw ten random sets and automatically evaluate these training sets on disjoint test sets from the same period. While the classification results vary only slightly between the ten different training sets, one would choose the model that yields the best results. Thus, the reported average values can be considered a conservative estimate.

We repeat this procedure for $k = (1, 2, ..., 22)$. As already observed in the cross validation, the best results can be achieved with a choice of $k = 2$: a detection rate of $\approx 94.2\%$ and false positive rate of $\approx 8.7\%$.

5.4 Feature Effectiveness

Approaches like EXPOSURE [5] that rely on a substantial amount of time-based features make it necessary to store each and every DNS answer observed in a network. Due to storage constraints, we store the data of each DNS answer containing an identical resource record only once. While this reduces the space complexity of

our approach, it renders the whole class of time-based features used by Bilge et al. unavailable to us. We faced a similar challenge with respect to features used in other approaches. Thus, we derived features from the available data to compensate for the unavailable features.

We also utilize the data used in Section 5.3 to evaluate the contribution each features makes to the classification process. Again, we use ten randomly drawn training sets and give the resulting detection and false positive rates as the average from these classifications. Additionally, we calculate the standard deviation as a measure of volatility of the classification accuracy with respect to the training set. As $k = 2$ has been shown to yield the best results, we compare classification results using kNN with $k = 2$. The *baseline feature set* contains the features 1, 4, 10, 12, and 13. This is the intersecting set of the feature sets used in previous work with the feature set that can be derived from the data available to us. Classification using only these features results in a detection rate of 58.86% and a false positive rate of 7.21%. However, the standard deviation of the detection rate is $\approx$ 27.41% and 2.72% for the false positive rate. This shows that the classification result is highly dependent on accidentally drawing a "good" random training set. We use this configuration as baseline to assess the contribution of each individual feature, i.e., we examine how the classification results change when we selectively add each novel feature or each group of features derived from the same kind of data. For instance, the bars labeled with *+consonantratio (2)* depict the detection and false positive rate of a classification using the baseline feature set plus feature 2, which is *consonantratio*. All results are shown in Fig. 3.

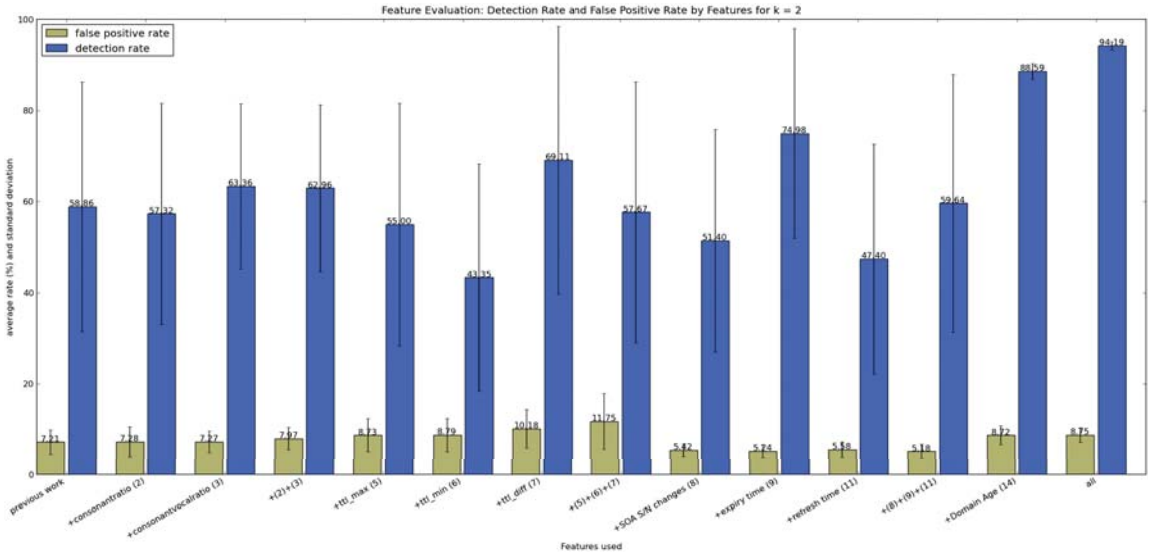


Fig. 3. Detection rate, false positive rate, and respective standard deviation by features used

The combination of all novel features added to the baseline feature set has a significant positive effect on the detection rate by increasing it from 58.86% to 94.19%. However, the newly introduced features also slightly increase the false positive rate by 1.54%. Yet, compared to the baseline set, these complete features are stable: when evaluating the same test set with ten different models based on ten randomly chosen training sets, the standard deviation is 1.64% for the false positive rate and only 0.86% for the detection rate while in the baseline feature set it is 2.72% and 27.41%, respectively. In summary, our features significantly improve the classification results.

6 Conclusion

We showed how data acquired via passive DNS replication and additional data like WHOIS and geolocation information can be used to identify botnet C&C domains. Based on our observations, we proposed 14 effective features to determine domain characteristics from sparse data. To the best of our knowledge, nine of these features have never been used before – thus provide an enlargement of the available feature choices. The proposed features can be divided into lexical features that are derived from the domain name itself, DNS-answer-based features that are directly derived from resource records, and IP address-based features like the host's geographic location and its location within the Internet's topology. Taking advantage of these features, we developed an approach to detect botnet C&C servers using machine learning methods. We tested the soundness of our approach on temporally disjoint training and test sets, a scenario as similar as possible to a real-life application environment of such a system. We found that our approach maintains a high detection rate of 94.2% and a relatively low false positive rate of 8.75%. Passive DNS replication is a very privacy-preserving technology, as only DNS answers are stored – information that is publicly available within the domain name system. At no point we make use of personally identifiable information. Our approach is thus fit to be used in a privacy-sensitive context.

Acknowledgements

This work has been supported by the Federal Ministry of Education and Research under BMBF Grant 01BY1111 (*MoBE*). We would like to thank Aaron Kaplan and the CERT.at team for providing us access to their passive DNS database.

References

1. abuse.ch. AMaDa, SpyEye and ZeuS C&C Domain Blocklists. `http://www.abuse.ch/`, 2012.
2. Alexa. Top 1,000,000 sites. `http://www.alexa.com/topsites`, July 2011.

3. Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *USENIX Security Symposium*, 2010.
4. Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon. Detecting malware domains at the upper DNS hierarchy. In *USENIX Security Symposium*, 2011.
5. Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: finding malicious domains using passive DNS analysis. In *Network and Distributed System Security Symposium*, 2011.
6. Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet(SRUTI)*, 2005.
7. Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *Proc. of the Third USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET)*, 2010.
8. Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *European Symposium on Research in Computer Security (ESORICS)*, 2005.
9. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition, 2009.
10. Yuanchen He, Zhenyu Zhong, Sven Krasser, and Yuchun Tang. Mining DNS for malicious domain registrations. In *Proc. of the 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2010.
11. Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C Freiling. Measuring and detecting Fast-Flux service networks. In *Network & Distributed System Security (NDSS)*, 2008.
12. Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.
13. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.
14. Jelena Mirkovic and Peter L. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *Computer Communication Review*, 34(2), 2004.
15. Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Internet Measurement Conference (IMC)*, 2006.
16. Florian Weimer. Passive DNS replication. In *17th Annual FIRST Conference on Computer Security*, 2005.
17. Sandeep Yadav and A.L. Narasimha Reddy. Winning with DNS failures: Strategies for faster botnet detection. In *Proc. of the 7th International ICST Conference on Security and Privacy in Communication Networks*, 2011.
18. Sandeep Yadav, Ashwath Kumar Krishna Reddy, A. L. Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proc. of the 10th annual conference on Internet measurement*, IMC '10, 2010.
19. Bojan Zdrnja, Nevil Brownlee, and Duane Wessels. DNSParse. `https://dnsparse.insec.auckland.ac.nz/dns/technical.htm`, 2007.

Statistical Modeling of Web Requests for Anomaly Detection in Web Applications

Harald Lampesberger[1], Markus Zeilinger[2], and Eckehard Hermann[2]

[1] Christian-Doppler Laboratory for Client-Centric Cloud Computing
Softwarepark 21, 4232 Hagenberg im Mühlkreis, Austria
h.lampesberger@cdcc.faw.jku.at
[2] University of Applied Sciences Upper Austria
Softwarepark 11, 4230 Hagenberg, Austria,
{firstname.lastname}@fh-hagenberg.at
http://www.fh-ooe.at

Abstract. We present an algorithm for learning a statistical representation of web application communication. The algorithm estimates the average probability of every observed web request. If the estimated probability deviates from recent observations, the web request is classified as anomalous. With every classification result, the statistical model parameters are updated, so the algorithm gains on-line learning capabilities and it self-adjusts to the observed data without prior training. Experiments on log data from a social networking web site indicate high detection rates at acceptable false-alarm rates. Also, the evaluation shows that the degree of abnormality of a web request does not explain the potential danger.

1 Introduction

Web applications traditionally use the Hypertext Transfer Protocol (HTTP) [6], an application layer network protocol, to connect client and server software running on different platforms. HTTP defines stateless and generic exchange of information between an initiating client and a server. The client requests a specific resource, identified by the so-called Unified Resource Identifier (URI), and the corresponding server answers with a status code, meta information and possible entity content. Today's web applications achieve session handling and dynamic content by evaluating specific parts of the request. Some prominent examples are:

- **Query in the Request-URI Path**: The path is a hierarchically structured sequence of string components with an optional query component. So-called *URL Rewriting* breaks this query-convention by encoding the query as string components to produce nice-looking URIs.
- **Request Headers**: meta data about the client or server, e.g. the so-called *cookie* header enables session tracking over the stateless HTTP.
- **Request Entity Content**: The client can use the HTTP POST or PUT method to send query-style or MIME-encoded data as request entity content.

A fundamental security problem is that the client is out of the server application's scope of control and cannot be forced to obey protocol specifications. If

client data is not handled correctly in any function of the web application, security weaknesses are introduced. Some weaknesses might be exploited by an attacker which leads to attack vectors like SQL/code injections, buffer overflows, cross-site scripting and many more [13]. Some examples are given in Figure 1.

```
Ok:  GET /fotos.php?action=view HTTP/1.1
Bad: GET /fotos.php?action=http://195.33.221.4:8081/bot.txt? HTTP/1.1
Bad: GET /userportal.php?id=4518-999.9+union+select+0-- HTTP/1.1
Bad: GET /fotos.php?action=search&album=%22%2F%3E%3Cscript%3Ealert
    %281%29%3B%3C%2FScript%3E HTTP/1.1
Bad: GET /images/../../../../../../../../../../../etc/passwd HTTP/1.1
```

Fig. 1. Examples for legitimate and malicious URI paths in HTTP requests.

1.1 Intrusion Detection

One approach in protecting web applications is to detect and prevent malicious client data by so-called intrusion detection systems (IDS). IDS can be distinguished into *misuse detection* and *anomaly detection* regarding their style of detection. While misuse detection relies on proper signatures of malicious behavior, anomaly detection tends to use methods such as machine learning or statistics to build a model of normal behavior and report deviating patterns as anomalies.

According to Sommer and Paxson [18], both concepts are challenged in different ways. The detection performance of misuse detection completely depends on currentness and coverage of signatures, but false-alarm rates are low. Anomaly detection is prone to costly false-alarms, but it is more likely to recognize novel attacks. Anomaly detection must especially consider a) the variability of input data, b) the lack of training data, c) a very high cost of errors, d) the difficulty of sound evaluation and e) descriptiveness of detection results.

1.2 Related Work

The payload-based anomaly detector (*PAYL*) by Wang and Stolfo [22] is the first published method for analyzing application data and it uses byte frequencies for payload profiling. *Anagram* [21] is an advancement of PAYL using n-grams instead of single byte frequencies. Perdisci et al. [15] present McPAD, a method based on 2_ν-grams to capture long range dependencies. The evaluations of PAYL, Anagram and McPAD include HTTP data. The first web-focused detection system is introduced by Kruegel and Vigna [9]. Their system combines six features like character distribution or attribute lengths to calculate an anomaly score. The work establishes a foundation for follow-up research: grouping similar anomalies [16], addressing concept drift [12] and dealing with scarce training data [17].

Ingham et al. [8] present an approach based on probabilistic finite automatons. Duessel et al. [4] introduce an attributed token kernel for One-Class Support Vector Machines to evaluate protocol syntax. Song et al. [19] propose a combination of multiple Markov chains named *Spectogram*. Krueger et al. [10] present a HTTP reverse proxy that detects and automatically repairs malicious web requests combining several detection methods.

The listed algorithms achieve good evaluation results, but they depend on training data. Especially for a fast-paced large-scale web application it can be impossible to create an representative training data set. Görnitz et al. [7] realize this problem and present an active learning strategy based on methods such as PAYL, Anagram and McPAD.

2 The Algorithm

The presented method builds upon two assumptions:

1. Valid web requests obey grammar G_{HTTP} and a specific web application effectively uses only a small subset of language: $L(G_{HTTP})$.
2. The prior probability of normal web requests is much higher than the probability of an attack.

This approach considers a web request to be a sequence of protocol elements. To cope with high-entropy elements, a web request is first transformed into a sequence of abstract symbols by splitting and counting character classes based on the G_{HTTP} specification. Furthermore, the language model is a statistical representation of symbol sequences considered normal for a specific web application. Inspired by Begleiter et al. [1], the language model estimates the probability of an observed web request of being normal. If the probability of the current sequence deviates too much from previously processed sequences, the current sequence is classified as either normal or anomalous and further actions, e.g. learning, are taken.

2.1 Request Transformation

Following RFC 2616 [6], several character classes are defined for G_{HTTP}. In a preprocessing step, we first use a modification of the *separator* class as defined in Equation 1 to tokenize the web request string. Distinguishing pre- and post-separators better preserves URI path characteristics in the tokens. An example is given in Figure 2.

$$\begin{aligned} \text{pre-separators} &= \{\texttt{SPACE TAB}\}, \\ \text{post-separators} &= \{\texttt{/?\&;()<>@,:"[]\{\}=\textbackslash}\}. \end{aligned} \tag{1}$$

In a second step, each token is reduced to an abstract symbol $\sigma \in \mathbb{N}^{16}$ by counting bytes that fall into classes as defined in Equation 2. A symbol σ is therefore a

statistical representation of bytes between two separators. This abstraction makes high-entropy strings like randomized identifiers comparable. After this transformation, a web request is turned into a sequence of symbols $q_{1:n} = (\sigma_1, \ldots, \sigma_n)$.

$$\sigma \Rightarrow \begin{cases} \sigma[0] & \text{number of printable ASCII characters,} \\ \sigma[1-4] & \text{lexical letter index} \in \{a..z, A..Z\} \mod 4, \\ \sigma[5-6] & \text{digit index} \in \{0..9\} \mod 2, \\ \sigma[7] & \text{capital letters} \in \{A..Z\}, \\ \sigma[8] & \text{lowercase letters} \in \{a..z\}, \\ \sigma[9] & \text{US-ASCII control characters,} \\ \sigma[10] & \text{protocol-specific bytes} \in \{\texttt{CR LF SPACE TAB}\}, \\ \sigma[11] & \text{path-specific characters} \in \{\texttt{./}\}, \\ \sigma[12] & \text{protocol separators} \in \{\texttt{?\&;()<>@,:[]\{\}=\textbackslash}\}, \\ \sigma[13] & \text{single and double quotes,} \\ \sigma[14] & \text{percent character,} \\ \sigma[15] & \text{non-US-ASCII character.} \end{cases} \tag{2}$$

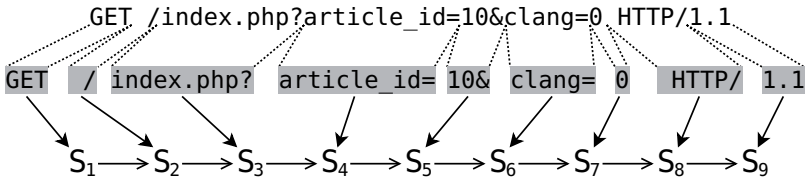


Fig. 2. Transformation of a web request data into a sequence of entities.

Our algorithm maintains a dynamic alphabet $\mathcal{A}$ which is the set of symbols the algorithm is aware of at a certain real-time moment. Initially, $\mathcal{A} = \emptyset$ and symbols are added and removed over time as the result of learning. Before the probability of a sequence can be evaluated, each symbol must be replaced with a similar one from the alphabet if possible. This requires a metric of similarity τ and a replacement function Φ. The chosen similarity measure is the Tanimoto coefficient:

$$\tau(\sigma_1, \sigma_2) = \frac{\sigma_1 \cdot \sigma_2}{\|\sigma_1\|^2 + \|\sigma_2\|^2 - \sigma_1 \cdot \sigma_2} \,. \tag{3}$$

Let $T_{\mathcal{A}}$ be threshold of similarity for alphabet $\mathcal{A}$ and also a model parameter. Two symbols σ_1 and σ_2 are considered identical if $\tau(\sigma_1, \sigma_2) > T_{\mathcal{A}}$. So, the mapping function Φ is defined as:

$$\Phi(\sigma, \mathcal{A}, T_{\mathcal{A}}) = \begin{cases} \arg\max\limits_{\nu \in \mathcal{A}} \tau(\sigma, \nu) & \text{if } \exists \nu \in \mathcal{A} : \ \tau(\sigma, \nu) > T_{\mathcal{A}}, \\ \sigma & \text{otherwise.} \end{cases} \tag{4}$$

2.2 Language Model

For probability estimation we use *Prediction by Partial Matching* (PPM) [2] as language model and Method-C [14] for smoothing zero-frequency counts. PPM-C is a variable-order Markov model (VMM) of maximum order D, a suitable data structure is a *trie* of depth $D+1$. Every trie node, except the root, references a symbol from alphabet $\mathcal{A}$ and maintains a frequency counter. Each path from root to node represents an observed subsequence during training and the node's count shows how many times the subsequence appeared. Figure 3 shows an exemplary trie for Markov order $D=2$.

In a sequence $q_{1:n} = (\sigma_1, \ldots, \sigma_n)$, the probability of each symbol q_i depends on its context $q_{i-D:i-1}$ of order D. The core idea of PPM-C is to back off to a smaller context if the count of symbol q_i in the context is zero. Estimating the probability $\hat{P}$ follows a recursive relation, where initially the current context order $k = D$ and $k < 0$ terminates the recursion:

$$\hat{P}(q_i|q_{i-k:i-1}) = \begin{cases} \hat{P}_k(q_i|q_{i-k:i-1}) & \text{if } q_{i-k:i} \text{ exists in trie,} \\ \hat{P}_k(escape|q_{i-k:i-1})\hat{P}(q_i|q_{i-k+1:i-1}) & \text{otherwise.} \end{cases} \tag{5}$$

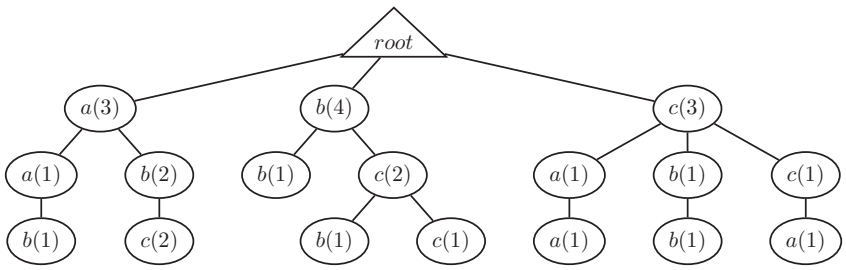


Fig. 3. PPM trie (order $D=2$) for sequence *abccaabcbb*.

Let $\mathcal{A}_s$ be the specific alphabet at context $s = q_{i-k:i-1}$ and $N(sq_i)$ be the count value of the node referencing symbol q_i in that context. Then the probability estimates based on Method-C are:

$$\hat{P}_k(q_i|s) = \frac{N(sq_i)}{|\mathcal{A}_s| + \sum_{\sigma \in \mathcal{A}_s} N(s\sigma)} \quad \text{if } q_i \in \mathcal{A}_s \ , \tag{6}$$

$$\hat{P}_k(escape|s) = \frac{|\mathcal{A}_s|}{|\mathcal{A}_s| + \sum_{\sigma \in \mathcal{A}_s} N(s\sigma)} \quad \text{otherwise.} \tag{7}$$

Finally, the average probability of a sequence $q_{1:n}$ is the arithmetic mean of its symbol probabilities:

$$\hat{P}(q_{1:n}) = \frac{1}{n} \sum_{i=1}^{n} \hat{P}(q_i | q_{i-D:i-1}) \; . \tag{8}$$

A VMM trained of all possible web requests delivers high mean probability scores for legitimate sequences. A malicious web request likely contains symbols that are unknown to the alphabet or in unexpected order. This results in a low mean probability of the sequence.

2.3 Classification

The distribution of sequence probabilities depends on the web application and its dynamics and a static threshold for classification of outliers is insufficient. We assume that mean sequence probabilities of legitimate web requests are quite similar and a *Beta* distribution can describe them. Unexpected web requests are accordingly in low density regions of the distribution and have low confidence. To estimate $Beta(\alpha, \beta)$, we first introduce real-time moments $t_1, \ldots, t_m$ that correspond to already symbolized web requests $q^{(1)}, \ldots, q^{(m)}$. Therefore at moment t_j, the historical mean $\bar{P}^{(j)}$ over the last w_{size} predictions and sample standard deviation $s^{(j)}$ are defined as:

$$\bar{P}^{(j)} = \frac{1}{w_{size}} \sum_{l=j-w_{size}}^{j} \hat{P}(q^{(l)}), \quad s^{(j)} = \sqrt{\frac{1}{w_{size}-1} \sum_{l=j-w_{size}}^{j} (\hat{P}(q^{(l)}) - \bar{P}^{(j)})^2}. \tag{9}$$

Using the method-of-moments [5, p. 40], the Beta distribution parameters $\hat{\alpha}^{(j)}$ and $\hat{\beta}^{(j)}$ at moment t_j are:

$$\hat{\alpha}^{(j)} = \bar{P}^{(j)} \left(\frac{\bar{P}^{(j)} \; (1 - \bar{P}^{(j)})}{s^{(j)^2}} - 1 \right), \quad \hat{\beta}^{(j)} = (1 - \bar{P}^{(j)}) \left(\frac{\bar{P}^{(j)} \; (1 - \bar{P}^{(j)})}{s^{(j)^2}} - 1 \right) \; . \tag{10}$$

The confidence $c^{(j)}$ of a web request $q^{(j)}$ is then estimated from the Beta cumulative distribution function:

$$c^{(j)} = I_{\hat{P}(q^{(j)})}(\hat{\alpha}^{(j)}, \hat{\beta}^{(j)}). \tag{11}$$

We define three confidence thresholds as model parameters: base confidence T_{base}, warn confidence T_{warn} and alert confidence T_{alert}. As a result, four confidence

intervals are formed in the distribution and Figure 4 outlines them. A web request is classified according to its confidence $c^{(j)}$:

$$classify(c^{(j)}) = \begin{cases} \textit{Normal} \text{ (learning)} & \text{if } c^{(j)} \geq 1 - T_{base}, \\ \textit{Normal} \text{ (ignore)} & \text{if } 1 - T_{base} > c^{(j)} \geq 1 - T_{warn}, \\ \textit{Anomalous} \text{ (warn)} & \text{if } 1 - T_{warn} > c^{(j)} \geq 1 - T_{alert}, \\ \textit{Anomalous} \text{ (alert)} & \text{otherwise.} \end{cases} \tag{12}$$

To sum up, a Beta probability distribution over the last w_{size} predictions enables classification of anomalous web requests without setting a static threshold for sequence probability. Depending on a web request's confidence, the grade of abnormality is known, it is assigned to one of four confidence intervals and further learning or reporting actions can be taken.

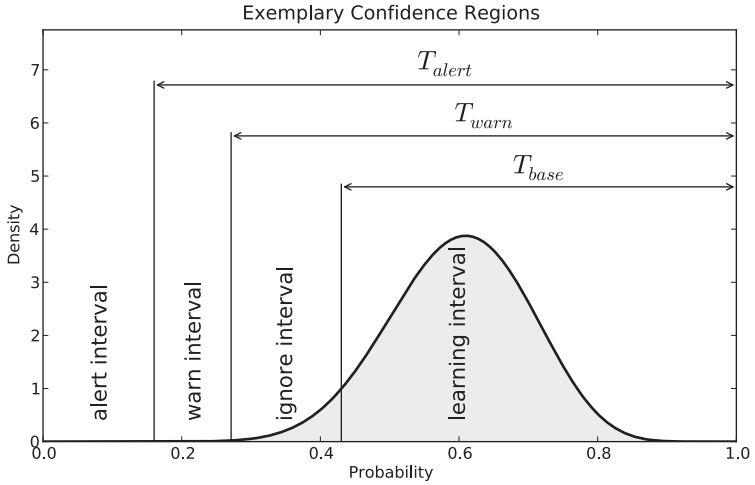


Fig. 4. Exemplary Beta probability-density function graph where the four confidence intervals (*alert*, *warn*, *ignore* and *learning interval*) are outlined.

2.4 Self-Adjustment and Learning

The language model requires updates to increase prediction precision and reduce PPM escapes over time. Better predictions result in higher mean $\bar{P}^{(j)}$ and lower sample variance $s^{(j)}$, the distribution and its confidence intervals get more and more distinct and detection performance improves. The on-line learning approach follows concepts of Vovk et al. [20].

Learning The first lazy teacher in the on-line learning scenario builds upon the assumption that normal web requests are much more likely than anomalous ones. Detections in the learning interval are automatically fed back into the language model and alphabet $\mathcal{A}$. The second lazy and slow teacher is a human expert who eventually recognizes a false positive or false negative with possible delay. In the case of a false positive, the sequence and novel symbols are added into the trie and alphabet. The according node counters are incremented until the sequence evaluates to the learning interval. If a false negative detection is corrected, the trie nodes related to the sequence are decremented or removed from the trie, where unreferenced alphabet symbols are also deleted.

Pruning Due to concept drift and numerical limits in computers, the trie and its counters cannot grow indefinitely. If the most frequent node count in the trie exceeds model parameter T_{prune}, pruning is performed. All node counters are integer divided by two, zero nodes or branches are removed and unreferenced symbols are deleted from the alphabet. So, escape probability mass increases again, the model is able to adapt to a certain degree of concept drift and malicious sequences learned by mistake will be dropped over time.

To sum up all introduced model parameters, the proposed anomaly detection model M is parameterized by:

$$M\langle T_{\mathcal{A}}, D, w_{size}, T_{base}, T_{warn}, T_{alert}, T_{prune}\rangle \ .$$

3 Performance Evaluation

For evaluation of detection performance, we assume a binary classification case where legitimate requests represent class *Normal* and warnings or alerts are considered as class *Attack*. A labeled data set is required to yield a confusion matrix as shown in Table 1. The values in the matrix are mandatory for estimating performance metrics.

Beside False Positive Rate (FPR), performance evaluation in this paper uses the metrics *Precision* and *Recall* as recommended by Davis and Goadrich [3] for skewed data sets:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}. \tag{13}$$

In PR space, a perfect algorithm has maximum Precision for the full range of Recall, the curve goes through the upper-right corner and PR-AUC = 1. The PR-AUC represents the capability of an algorithm to correctly separate the two classes in the binary classification case.

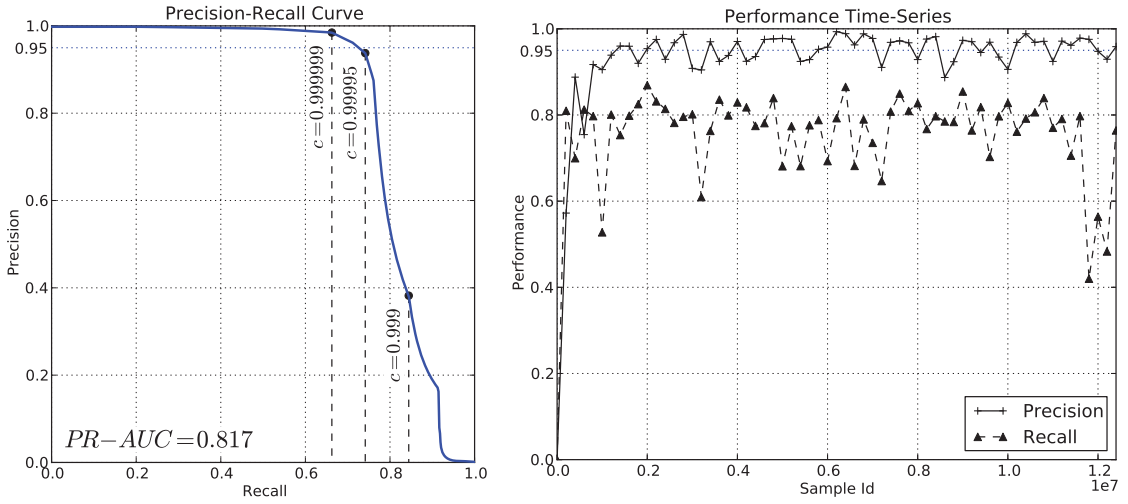
Table 1. Confusion matrix for the binary classification case.

		Actual	
		Attack	**Normal**
Predicted	**Attack**	True Positive (TP)	False Positive (FP)
	Normal	False Negative (FN)	True Negative (TN)

3.1 The SOCIAL Data Set

Sound evaluation requires realistic data. The presented evaluation is based on log files from a social networking web site because a log line contains parts of the web request and is therefore sufficient for basic experiments. The web site combines several web applications and is a worst-case for intrusion detection. The 12.5M log lines were annotated manually to establish ground truth and instances of a pool of 57 attacks were inserted randomly. The resulting data set has 12,528,513 samples where a total of 14,465 are considered anomalous. During the evaluation, a virtual expert randomly gives feedback to the algorithm for 10% of false negatives and 66.6% of false positives.

A parameterized model $\langle 0.8, 4, 20000, 0.995, 0.99995, 0.99995, 5000000 \rangle$ yields the best performance with Recall = 74.15% and Precision = 93.76% and detection results are shown in Figure 5. A total of 714 false positives cause FPR $= 5.71 \cdot 10^{-5}$. The two least-recalled classes of annotated anomalies are non-critical scanning attempts and an application-specific JavaScript fault.



(a) Overall PR curve. (b) PR time-series curve.

Fig. 5. Performance metrics for the SOCIAL data set.

Figure 5(b) is a time-series of Precision and Recall. It clearly shows that a Precision $\geq$ 90% is reached after about 1M processed samples. The presented algorithm

self-adjusts to the observed data. At the real-time moment of the last processed sample, the language model has 19,650 trie nodes, an alphabet size $|\mathcal{A}| = 100$ and permits throughput of 29,200 logs/second.

4 Conclusion

The contribution of this article is an algorithm for anomaly detection in web applications and its evaluation. Experiments indicate decent results and an extended version of this article is available in [11].

The evaluation has shown that the binary classification case is insufficient for realistic scenarios. The confidence of a detection reflects its abnormality at that specific real-time moment of the algorithm but it does not explain the impact or dangerousness of the detected anomaly. Also for future research, the shift from web applications to web services must be considered for intrusion detection.

References

1. R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *J. Artif. Int. Res.*, 22(1):385–421, 2004.
2. J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402, 1984.
3. J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.
4. P. Duessel, C. Gehl, P. Laskov, and K. Rieck. Incorporation of application layer protocol syntax into anomaly detection. In *ICISS '08: Proceedings of the 4th International Conference on Information Systems Security*, pages 188–202, Berlin, Heidelberg, 2008. Springer.
5. M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions, 3rd Edition*. Wiley-Interscience, 2000.
6. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785.
7. N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. Active learning for network intrusion detection. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, AISec '09, pages 47–54, New York, NY, USA, 2009. ACM.
8. K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest. Learning dfa representations of http for protecting web applications. *Comput. Netw.*, 51:1239–1255, April 2007.
9. C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261, New York, NY, USA, 2003. ACM.
10. T. Krueger, C. Gehl, K. Rieck, and P. Laskov. Tokdoc: a self-healing web application firewall. In *SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1846–1853, New York, NY, USA, 2010. ACM.
11. H. Lampesberger, P. Winter, M. Zeilinger, and E. Hermann. An on-line learning statistical model to detect malicious web requests. In *Security and Privacy in Communication Networks - 7th International ICST Conference, SecureComm 2011*, London, 2011. Springer.
12. F. Maggi, W. Robertson, C. Kruegel, and G. Vigna. Protecting a moving target: Addressing web application concept drift. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, Saint-Malo, France, September 2009.

13. MITRE Corporation. Common Vulnerabilites and Exposures. `http://cve.mitre.org/`, 2012. [Online; accessed 12-Mar-2012].
14. A. Moffat. Implementing the ppm data compression scheme. *Communications, IEEE Transactions on*, 38(11):1917 –1921, November 1990.
15. R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864 – 881, 2009. Traffic Classification and Its Applications to Modern Networks.
16. W. Robertson, G. Vigna, C. Kruegel, and R. Kemmerer. Using generalization and characterization techniques in the anomaly-based detection of web attacks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2006.
17. W. Robertson, F. Maggi, C. Kruegel, and G. Vigna. Effective anomaly detection with scarce training data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2010.
18. R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. *IEEE Symposium on Security and Privacy*, pages 305–316, 2010.
19. Y. Song, A. D. Keromytis, and S. J. Stolfo. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.
20. V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer New York, Inc., Secaucus, NJ, USA, 2005.
21. K. Wang, J. J. Parekh, and S. J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Advances in Intrusion Detection*, volume 4219 of *LNCS*, pages 226–248. Springer, 2006.
22. K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, volume 3224 of *LNCS*, pages 203–222. Springer, 2004.

Automatic Generation of Generalizing Behavioral Signatures for Early Warning Systems

Martin Apel and Michael Meier

[1] Technische Universität Dortmund
Information Systems and Security, 44221 Dortmund
[2] Fraunhofer FKIE, Cyber Defense
Neuenahrer Str. 20, 53343 Wachtberg

Abstract. Malware poses a major threat on the Internet today. Polymorphic obfuscation techniques employed by malware render static malware analysis and detection approaches based on static signatures ineffective. We present our malware early warning system that is based on dynamic behavioral analysis and aims at deriving generalizing behavioral detection signatures for malware. In the context of early warning a particularly desired property of the generated signatures is not only to detect the malware binaries which are known during signature generation, but also to detect unknown (polymorphic) variants of these known binaries. We present a tentative experimental study on the generalization of behavioral signatures and investigate the question, which share of a polymorphic malware family is required to be known for generating a signature showing an acceptable detection performance for the whole family? Experimental results showed that for 90% of the considered malware families less than 14 family members are required to generate a behavioral signature that detects at least 80% of the whole family. The results are quite promising and encourage further investigation of our approach for automatic generation of behavioral detection signatures.

1 Introduction

Though the term Early Warning System (EWS) has been used in the literature, there is no common, accepted definition of what early warning is (and no differentiation to, e.g., intrusion detection systems (IDS)). A vague definition of an early warning information system (EWIS) is given by [6], who defines EWIS by sketching its purpose: *'EWIS assists experts and policy makers in assessing desired options for...'* several particular security measures. By outlining one particular realization technique, [6] defines *'[an EWS] for IT security surveillance based on a specific procedure to detect as early as possible any deviation from usual or normal observed frequency of phenomena.'* A more general operational definition of early warning is used by [8]: *'In case of perceptible indicators, and no or (still) a low number of victims, information must be distributed to help others - not yet victims including response organizations in order to avoid a major crisis.'* We adopt the more declarative definition of [3]: *'EWS aim at detecting unclassified but potentially harmful system behavior based on preliminary indications and are complementary to intrusion detection systems. Both kinds of systems try to detect, identify, and react before*

possible damage occurs and contribute to an integrated and aggregated situation report (big picture). A particular emphasis of EWS is to establish hypotheses and predictions as well as to generate advices in still not completely understood situations. Thus the term early has two meanings, a) to start early in time aiming to avoid/minimize damage, and b) to process uncertain and incomplete information.' In this paper we consider an instance of an EWS which is particularly targeting the malware threat and explore its ability to detect unknown malware (variants) based on incomplete information.

Malware poses a major threat on the Internet today and is defined as software performing actions intended by an attacker without consent of the computer's owner when executed. Forms of malware include worms, viruses, Trojan horses, bots, spy- and adware. Driven by commercially oriented criminals which systematically compromise computers connected to the Internet for illegal purposes, e.g. distribution of spam messages, the malware threat has grown dramatically.

Collecting novel malware binaries in principal allows one to develop protective anti-malware measures, such as signatures for anti-virus scanners. Several approaches exist for automatic collection of novel malware binaries [7,4], which are typically based on honeypots. However, more than 55.000 novel malware binaries per day [9] challenges developers of anti-malware-products who thus need to be supported by automatic malware analysis techniques.

Static analysis techniques (e.g. [5,13,14]) focus on static features of malicious binaries. Malware authors typically employ obfuscation engines like ADMutate [10] which pack or encrypt binaries in order to morph malware binaries during distribution to avoid detection by static signatures. This leads to populations of numerous polymorphic variants of a malware requiring a static signature for every variant to detect the whole population. Owing to these facts the effectiveness of automated static analysis of malicious binaries is limited.

In contrast, dynamic analysis is based on monitoring the behavior of malware during execution of malicious binaries, e.g. the sequence of performed system calls, which is more difficult to conceal or obfuscate. Since all polymorphic variants of a malware population realize the same functionality, they show similar behavior. Therefore dynamic analysis recently became popular and forms the basis of many approaches for automatic analysis of malware binaries (e.g. [18,11]).

Given their behavior malware binaries can be clustered into groups of similar behavior, which are referred to as malware families. Particularly, polymorphic variants of a malware that share malware-specific behavior can be grouped by this approach. Being able to automatically cluster malicious binaries into families of similar behavior provides malware analysts with two opportunities. First, every time a new malware binary is found it can be quickly determined whether it is a variant of an already known malware family or a new one. Second, and going one step further, automatic behavioral clustering of malware is a starting point for automatic cre-

ation of generalizing behavioral signatures which are intended to be included in our malware early warning system [1]. Behavior that is shared by all malware binaries of a particular cluster can be used to create a generalizing behavioral signature which matches all malware binaries of this cluster. Of course, this is only one step and not sufficient, since it must be ensured that the signatures do not match any behavior of benign programs.

Besides providing a situation picture our malware early warning system aims at deriving generalizing behavioral detection signatures for malware. These signatures are generated by combining the following technologies to an integrated process chain: 1) capturing actively spreading malware using honeypot-based malware collectors, 2) behavioral analysis of malware binaries generating behavior reports for each malware binary, 3) grouping of behavior reports into groups of similar reports, and 4) generating signatures for each group of behavior reports.

In the context of early warning a particularly desired property of the generated signatures is not only to detect the (behavior of) binaries which are known during signature generation, but also to detect (polymorphic) variants of these known binaries. Ideally a signature generated for a few binaries with similar behavior should detect all binaries showing similar behavior. I.e. a signature generated based on a few captured polymorphic variants of a malware should detect all variants of this malware thus making it a generalizing signature. Based on our experience with the malware early warning system prototype we study the generalization aspect of generated behavioral signatures and experimentally investigate the question, which share of a malware family is required to be known for generating a signature showing an acceptable detection performance for the whole family?

The paper is structured as follows. Section 2 gives an overview of our malware early warning system. Our experience with the system, the experiments conducted and the results of those experiments are described in Section 3. Some final remarks on the applicability of early warning conclude the paper.

2 Behavioral Malware Signature Generation

In the considered framework of behavioral signature generation a given set of malware binaries is processed by successively applying the following four automatic procedures. First, behavioral features are extracted from each malware binary and put in a representation called behavior report. Second, distances between all behavior reports are measured. Third, the behavior reports are grouped in reports with small distances using clustering. Fourth, based on features common to all behavior reports in a group signatures are created. Each of the four procedures is shortly described in the following.

2.1 Extracting and Representing Features

The behavior of a program is defined by the sequence of interactions with (objects of) the execution environment during program execution, i.e. the sequence of executed system-calls. To allow an automatic analysis of program behavior it needs to be captured by some monitoring mechanisms which logs executed instructions in their order of execution. For capturing the behavior of a malware binary, it is executed in a controlled monitored environment using CWSandbox [18], a tool specifically developed for dynamic analysis of malware. We discuss the properties of this behavior monitoring approach in the following. The execution of a Windows

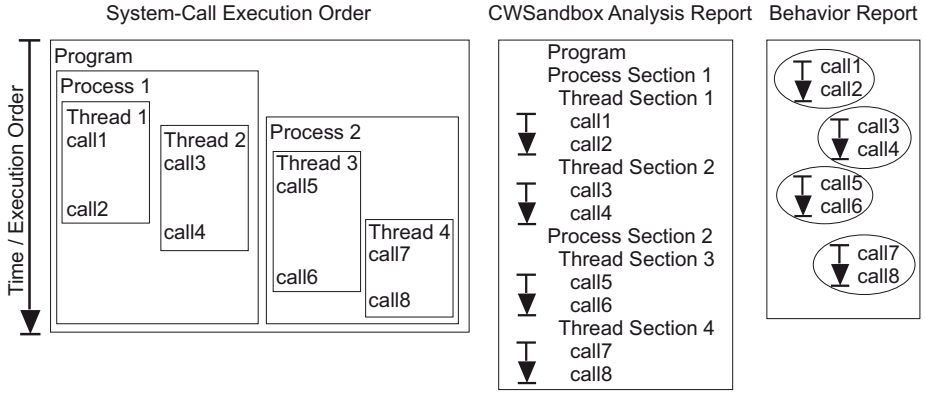


Fig. 1. Real and represented order of execution

program is performed by at least one process that contains at least one thread. While a process defines the execution context that is shared by all its threads, a thread represents the execution of an instruction sequence. During execution a program can create additional processes and threads. If not explicitly synchronized all threads of a program are executed concurrently (see Fig. 1).

Monitoring program execution using CWSandbox results in an analysis report containing a process section for each process which is created during program execution. Each process section contains a section for each thread created by the respective process during execution. A thread section contains a chronologically ordered list of the system-calls executed by the thread. Due to concurrent execution all the process sections of the analysis report and the thread sections of a process section are not chronologically but arbitrarily ordered (see Fig. 1). Consequently, the system-call execution order is captured only thread-local. The only inter-thread or inter-process relation captured is the parent-child relationship, i.e. which thread (resp. process) has created another thread (or process).

CWSandbox monitors program behavior at system-call level. There are different Windows32 system-calls that can be used to realize the same functional-

ity. CWSandbox maps all system-calls realizing a particular functionality onto a functionality-specific CWSandbox function (CWS function). For example, several system-calls exist to determine the current system time. All these system-calls are logged by CWSandbox as *get_system_time*. The particular system-call is given as parameter in the log entry. CWSandbox (version 2.0.71) knows about 130 CWS functions. Repetitive executions of a system-call with the same parameters are aggregated by CWSandbox and logged as one system-call. The particular number of calls is given as parameter in the log entry. For any executed system-call, CWSandbox creates a log entry containing the respective CWS function, the system-call arguments, the system-call result, the name of the particular system-call, and the number of aggregated repetitive executions of the system-call.

CWSandbox uses an XML-based format to store CWSandbox analysis reports. For effective and efficient analysis, the syntactical data representation needs to be adjusted such that discriminative data feature patterns are accessible by the analysis methods. Therefore we preprocess and transform CWSandbox analysis reports into so-called behavior reports. This transformation discards any analysis meta-information contained in the CWSandbox analysis report. It also discards any arguments to and results of system-calls and the number of aggregated repetitive executions of system-calls. While discarding system-call arguments may render the approach vulnerable to mimicry attacks (see e.g. [15]). We however see the system-call itself as most important behavioral feature to focus on. If the proposed approach performs well on these features we plan to extend it in future work, e.g. by identifying suitable system-call arguments. Further, the transformation ignores the parent-child relationship between threads and processes. Considering the chronologically ordered sequences of CWS function calls of thread sections as strings over the alphabet of CWS functions, the resulting behavior report constitutes a set of strings characterizing the behavior of threads. Thus any order between the thread sections is ignored (see Fig. 1).

The discussion in following sections is based on the concept of *shared behavior* of several behavior reports, which is defined as follows: A string s is called a common substring of a set of behavior reports $\{R^1, R^2, ..., R^m\}$ with $R^i = \{r_1^i, r_2^i, ...\}$ iff for each behavior report R^i there exists a thread denoted by j_i such that s is a common substring of the strings $r_{j_1}^1, r_{j_2}^2, ...$ and $r_{j_m}^m$. The *shared behavior* of a set of such behavior reports is then defined by the set of maximal common substrings of the behavior reports. For example, given a set of behavior reports $S = \{R^1, R^2, R^3\}$ with $R^1 = \{abcd, xyz\}, R^2 = \{bcda, yz\}$ and $R^3 = \{zx, bca\}$, the shared behavior of S is the set of strings $\{a, bc, z\}$.

2.2 Measuring Similarities between Behavior Reports

To determine whether two behavior reports are similar a distance between them is calculated. For this several distance measures can be used. We studied different

distance measures in detail and discussed desirable properties of a distance measure for this particular purpose in [2]. The considered distance measures include the edit distance (also known as Levenshtein distance), an approximated edit distance, the normalized compression distance, and the Manhattan distance. They were evaluated with respect to the criteria appropriateness, order sensitiveness, and run-time performance. Appropriateness of a distance measure reflects the fact that behavior reports which share a large amount of behavior have a small distance and that behavior reports, which share a tiny (or no) amount of behavior, have a large distance. For comparative evaluation of their appropriateness, the distance measures were rated according to the amount of shared behavior of groups (or clusters) which can be found after clustering behavior reports using a particular distance measure. Regarding order sensitiveness, a distance measure is considered appropriate if it is sensitive to changes at a local level (e.g. the sequence of system-call executions needed to perform a specific task), but insensitive to changes at a global level (e.g. the sequence in which different tasks are accomplished). Based on the results of this study, we identified the Manhattan distance as the most appropriate distance measure for grouping malware behavior reports with similar behavior. To apply the Manhattan distance the behavior reports have to be embedded into a vector space first. This is done using a formal language L. Each dimension of a vector corresponds to a specific word of L and refers to the number of occurrences of the word in the behavior report. The vector dimension depends on the number of words in L, which is usually very high. On the other hand the vectors are very sparse. Data structures tailored for such vectors, e.g. arrays, tries and suffix trees, are described in [16]. We use n-grams of system-calls as the formal language to embed behavior reports into a vector space. According to our analysis results in [2], n has been chosen to be 3.

2.3 Clustering Behavior Reports and Generating Behavioral Signatures

A number of proposals to cluster malware behavior have already been made in the literature (e.g. [12,17]). Given a distance measure for behavior reports, clustering approaches can be used to group close (short distant) behavior reports into so-called behavior clusters. Unfortunately, there is no fixed/distinct distance known up to which behavior reports should belong to the same cluster. Therefore we need an additional criterion for deciding whether a cluster should be formed or not. Since it is the goal to form signatures for every cluster, we use the existence of a good signature as criterion to decide whether a cluster is formed or not. A signature is considered good if it does not match the behavior of a set of benign programs (the so-called Goodpool).

Because of the incremental nature of the problem (binaries and thus behavior reports arrive continuously) an incremental clustering approach is needed.

1. The algorithm tries to insert the new behavior report into existing clusters without changing their signatures first.
2. If that does not work the algorithm tries to insert the behavior report into the "best fitting" (minimal distant) cluster (the distance measure is used as a heuristic here) so that the modified signature for that cluster is still valid (does not match behavior reports of the Goodpool).
3. If step 2 fails too, a new cluster for the behavior report is created.

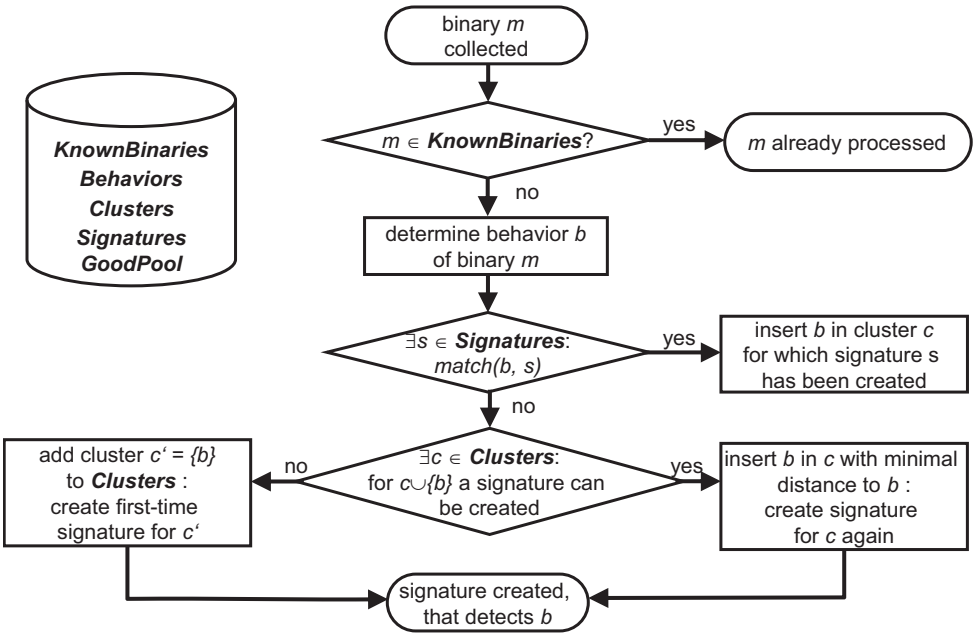


Fig. 2. Incremental clustering and signature generation for behavior reports

A sketch of the overall procedure can be found in Fig. 2. The algorithm ensures that clusters are containing as much reports as possible without creating an invalid signature for the clusters. The generated set of clusters and signatures will cover the given malware behavior reports, but it does not match any behavior report in the Goodpool.

3 Experience and Experiments

We have implemented the procedures described above in our malware early warning system prototype which runs since September 2009. It uses the server honeypot Amun [7] as malware collectors and starting from June 2010 drive-by-downloaded binaries collected by the Web Honeyclient MonkeyWrench [4] are also forwarded to

our system. Up to now (April 2012) about 106.910.920 attacks have been registered. Overall 35.460 unique malicious binaries have been submitted to our system. About 15.000 of these binaries could be analyzed successfully using CWSandbox. The transformation of the respective CWSandbox analysis reports into behavior reports resulted in 12.839 unique behavior reports. These behavior reports are spread over 171 clusters. The Goodpool currently used by our system contains behavior reports of 1.500 benign programs - most of them are installation programs of freely available software packages.

Based on the experience and the achieved results we are studying the generalization aspect of generated behavioral signatures. The question is investigated, which share of a malware family is required to be known for generating a signature showing an acceptable detection performance on the whole family? To prevent a single behavior report dominating the signature generation and thus the generalization process for the whole cluster, only clusters that contain at least ten behavior reports are used in the experiment. This leaves us with 50 clusters of different sizes. The number of clusters of the different sizes is shown below in Fig. 3. Note that

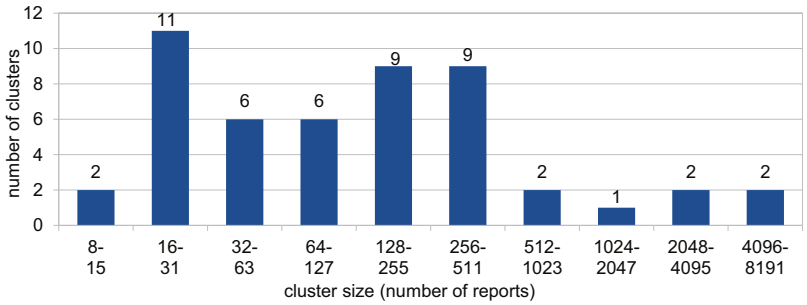


Fig. 3. Number of clusters of different sizes

the interval size doubles with every step, but the number of clusters stays roughly the same, i.e. the number of small clusters is large compared to the number of large clusters. For signature generation in an early warning context, two related questions are of particular importance: 1. How fast does the true positive rate increase with respect to the number of known behavior reports used for generating the signature? 2. How many behavior reports need to be seen and processed before a reasonable signature can be created?

To answer the first question for a certain cluster we measured the true positive rate for signatures which are created using randomly drawn behavior reports from this cluster. This was repeated five times for each number of behavior reports. The arithmetic mean of the measured true positive rates was used to generate a graph for each cluster showing the averaged true positive rate relative to the number of reports

used for signature generation. The resulting graphs for a medium size cluster of 303 behavior reports are shown in Fig. 4. For answering the second question we evaluated

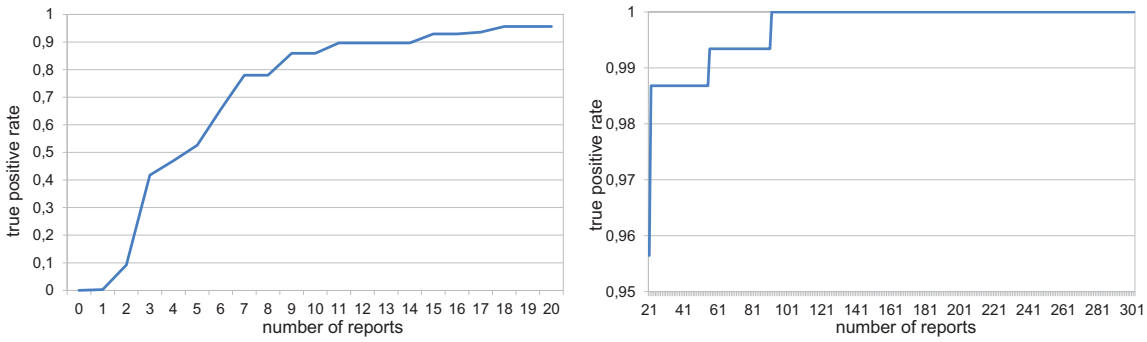


Fig. 4. : True positive rate per number of behavior reports used for signature generations (medium size cluster)

the number of behavior reports required for generating a signature achieving a true positive rate of at least 80%. The results are shown in Table 1 and Fig. 5. They point out that for 90% of the clusters 13 behavior reports from each cluster are enough to generate a signature detecting at least 80% of the reports within the clusters. Fig. 5 suggest that there is no direct obvious connection between cluster

Num. of behavior reports required for generating a signature with true positive rate $\geq$ 80%	Num. of clusters	Sizes of corresponding clusters
3	1	16
5	2	17, 28
6	6	13, 15, 379, 409, 922, 6513
7	5	23, 25, 70, 338, 375
8	7	19, 21, 49, 123, 186, 446, 5827
9	5	24, 56, 62, 303, 2446
10	5	32, 44, 143, 168, 880
11	5	42, 103, 139, 207, 1895
12	3	26, 74, 136
13	6	24, 74, 166, 213, 435, 2802
15	2	122, 430
16	1	201
25	1	472
30	1	30

Table 1. Number of behavior reports required for generating signatures with a detection rate of at least 80%

size and the number of reports that are needed to achieve 80% detection rate. This is quite surprising. Whether this effect is only present in our dataset or is a general

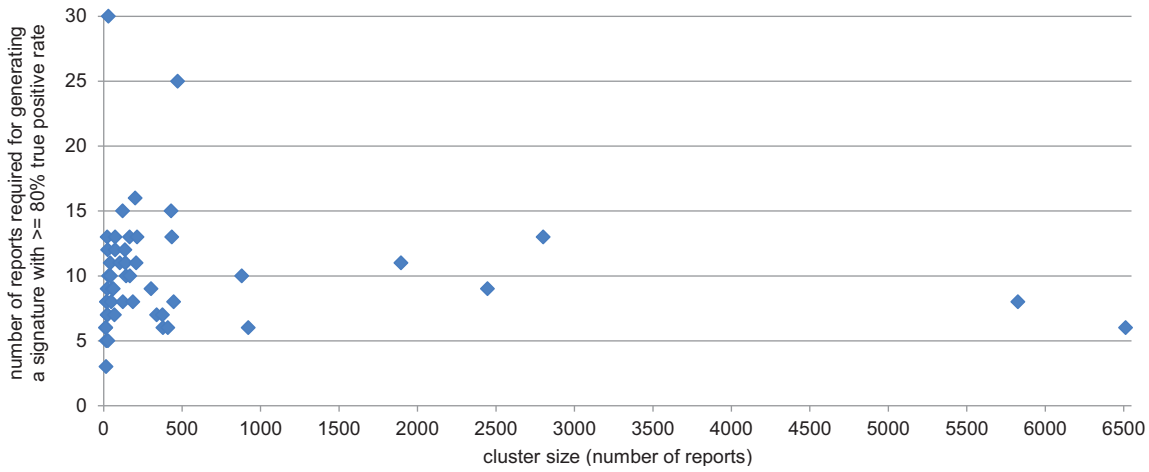


Fig. 5. Number of behavior reports required for generating signatures with a detection rate of at least 80% versus cluster size

phenomenon needs to be investigated further. The described experiments show that generalization effects happen quite fast, i.e. only a few behavior reports are required to generate useful generalizing signatures.

4 Final Remarks

Our system is an example of an early warning system which aims at deriving generalizing detection signatures for malware. Based on incomplete information (only a few behavior reports) about a malware behavior cluster (family), the system automatically generates behavioral signatures that have a reasonable detection rate with respect to the overall cluster. In this paper we studied the generalization aspect of generated behavioral signatures and experimentally investigated the question, which share of a malware family is required to be known for generating a signature showing an acceptable detection performance for the whole family?

Experimental results show that for 90% of the considered clusters less than 14 behavior reports from a cluster are required to generate a signature that detects at least 80% of the behavior reports in these clusters. Our results also show that the detection rate of a generated signature is increasing very rapidly with the amount of behavior reports used for its generation. This makes them particularly usable for the early warning context, where fast responses (signatures) are required based on incomplete information.

The behavioral signature generation approach also seems to be suitable for thwarting the threat posed by polymorphic transformations incorporated in malware, because signatures which can detect a large percentage of a population of polymorphic malware variants can already be generated after only a few of them

are known. To sum up, the described results of this tentative experimental study about the generalization of behavioral signatures encourage further detailed investigations of the behavioral signature generation approach in future work.

References

1. M. Apel, J. Biskup, U. Flegel, and M. Meier. Towards Early Warning Systems – Challenges, Technologies and Architecture. In *Proc. of CRITIS 2009*, volume 6027 of *LNCS*, pages 151–164. Springer, 2009.
2. M. Apel, C. Bockermann, and M. Meier. Measuring similarity of malware behavior. In *Proc. of 34th LCN 2009*. IEEE Computer Society, 2009.
3. J. Biskup, B. M. Hämmerli, M. Meier, S. Schmerl, J. Tölle, and M. Vogel. 08102 working group – early warning systems. In *Perspectives Workshop: Network Attack Detection and Defense*, volume 08102 of *Dagstuhl Seminar Proceedings*, 2008.
4. A. Buescher, M. Meier, and R. Benzmueller. Throwing a monkeywrench into web attackers plans. In *Proc. of the 11th Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security (CMS 2010)*, volume 6109 of *LNCS*, pages 28–39. Springer, 2010.
5. M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proc. of the 12th USENIX Security Symposium*.
6. U. Gattiker. *The Information Security Dictionary*. Kluwer, 2004.
7. J. Göbel. Amun: Automatic Capturing of Malicious Software. In *Proc. of Sicherheit 2010*, volume 170 of *LNI*, pages 177–190. GI e.V., 2010.
8. B. Grobauer, J. Mehlau, and J. Sander. Carmentis: A co-operative approach towards situation awareness and early warning for the internet. In *Proc. of IMF'06*, volume 97 of *LNI*, pages 55–66. GI, 2006.
9. AV-Test Institute. AV-Test - Malware Statistics. `http://www.av-test.org/en/statistics/malware/`, Last accessed 30 April 2012.
10. K2. Admmutate 0.8.4. `http://www.ktwo.ca/security.html`, Last accessed 30 April 2012.
11. A. Lanzi, M. Sharif, , and W. Lee. A system for extracting kernel malware behavior. In *Proc. of NDSS'09*, 2009.
12. T. Lee and J. J. Mody. Behavioral classification. In *Proc. of Annual Conference of the European Institute for Computer Antivirus Research (EICAR)*, 2006.
13. C. Linn and S. Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In *Proc. of Conference on Computer and Communications Security (CCS03)*, pages 290–299. ACM, 2003.
14. A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *Proc. of the sss Annual Computer Security Applications Conference (ACSAC 2007)*, pages 421–430. IEEE Computer Society, 2007.
15. C. Parampalli, R. Sekar, and R. Johnson. A practical mimicry attack against powerful system-call monitors. In *Proc. of the 2008 ACM symposium on Information, computer and communications security*, ASIACCS '08, pages 156–167, New York, NY, USA, 2008. ACM.
16. K. Rieck and P. Laskov. Linear-Time Computation of Similarity Measures for Sequential Data. *Journal of Machine Learning Research (JMLR)*, 9:23–48, 2008.
17. P. Trinius, C. Willems, T. Holz, and K. Rieck. Automatic Analysis of Malware Behavior using Machine Learning. *Journal of Computer Security (JCS)*, 19(4):639–668, 2011.
18. C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using CWSandbox. *IEEE Security & Privacy*, 5(2):32–39, 2007.

A Concept for Secure and Privacy-Preserving Collaborative Information Sharing

Hans Hofinger and Sascha Todt

Fraunhofer AISEC,
Parkring 4, 85748 Garching (near Munich), Germany
hans.hofinger@aisec.fraunhofer.de,sascha.todt@gmx.de
http://www.aisec.fraunhofer.de

Abstract. The enormous number of different threats makes it very hard for network operators to detect, analyze and counter attacks on systems within their administrative domain. By using an IT Early Warning System to collaborate they may improve attack detection and analysis results and reduce the time and costs spent for incident handling. To enable collaboration, participants of an information sharing network need to exchange security relevant data which raises several considerations regarding privacy and reputation. To eliminate these issues and encourage collaboration we present a concept for secure and privacy-preserving information sharing across borders of administrative domains and show how existing techniques can be used to fulfill the manifold requirements of IT Early Warning.

1 Introduction

The ultimate goal of IT Early Warning (IT-EW) is to identify attacks and threats to communication networks at a very early stage and to provide information about these incidents. This enables organizations to take countermeasures and minimize or even avert the negative impacts of such attacks. The type of information needed, however, highly depends on the type of attack that shall be detected. While the detection of some attacks requires the analysis of data from merely one administrative domain it might also be necessary to collect data from more than one organization to provide the necessary base to identify (large-scale) attacks or their indications.

Hence from a technical perspective it is advantageous for different network operators to collaborate and share security incident related information between each other to better be able to react and mitigate attacks on or emerging from their infrastructures.

In the scenario where information is required from multiple administrative domains it might be necessary that an entity shares information that contains sensitive data. This, however, raises data privacy issues since sharing such information with other organizations, especially if those are competitors, may have negative impact on the organization's future business and success or might be prohibited by law. Hence there will be reluctance to provide this information to a collaborative IT Early Warning System (IT-EWS). Thus, it is essential to build privacy-preserving

techniques into such systems. These techniques shall not only guarantee data privacy but also originator anonymity to assure that no conclusions can be drawn about sensitive data or the initial transmitter of a piece of information.

Many IT-EWSs rely on a trusted central instance for analysis and correlation of the data that is collected at various administrative domains [6,11] . This, however, does not solve all of the data privacy issues, since some types of data still have to be sanitized or pseudonymized (as done in, e.g., [13]) due to law or protection of assets before sent to a foreign organization. Additionally some potential users of an IT-EWS may be reluctant to trust a third party at all for various reasons. In that case there is a need for a decentralized system approach and sophisticated detection and analysis techniques, that can be built on top of this system.

Such a decentralized approach is investigated as one of the main parts of the ASMONIA[1] project which is sponsored by the German Federal Ministry of Education and Research[2] (BMBF). The main goal of this project is to improve the security for next generation mobile communication networks, such as 4G/LTE.

Similar to other techniques like firewalls or intrusion detection and prevention systems that aim to mitigate attacks an adversary might try to circumvent or even attack an IT-EWS itself to obfuscate her activities. Thus, further security considerations have to be taken into account in addition to privacy and anonymity issues that have to be addressed. These considerations range from vulnerability to (distributed) denial of service attacks to injection of falsified data to fool participants of an IT-EWS.

Thus, our approach, named Collaborative Resilient Early Warning (CREW), provides a distributed, decentralized, resilient, anonymity and privacy-preserving information sharing platform which aims for improving the

- accuracy and timeliness of warnings
- detection and analysis of attacks and anomalies
- resilience to attacks against an IT-EWS itself.

2 Challenges in Collaborative Information Sharing

Collaboration between different organizations comes along with a number of issues regarding privacy, loss of reputation and competitive disadvantages, especially if participants are competitors. Thus, this chapter outlines security and privacy considerations for participants of an IT-EWS and discusses requirements for a system that enables collaboration between them.

An IT-EWS has to ensure, e.g., that the originator of a report containing meaningful warning information for other participants must not be afraid that its real

[1] http://www.asmonia.de
[2] http://www.bmbf.bund.de

identity can be inferred from, e.g., the source IP address of the warning or the report data itself. Nevertheless it must be possible to trace the sources of purposely distributed wrong information which may lead to negative impacts at the recipients' networks. Obviously this leads to a conflict of requirements. Our approach minimizes this conflict and provides a solution that fulfills the requirements as good as possible.

The following list summarizes the requirements we have identified for a reliable and privacy-preserving collaborative information sharing system protecting the participants' reputation:

Anonymity to prevent loss of reputation for participants sharing sensitive information

Privacy to prevent competitive disadvantages for participants sharing sensitive information

Authentication to form a closed user group of legitimate participants and avoid injection of falsified information by an attacker

Integrity to guarantee that information sent via the IT-EWS cannot be modified unnoticed between sender and receiver

Confidentiality to guarantee the privacy of information exchanged via the IT-EWS

Non-repudiation to enable traceability of falsified messages injected into the IT-EWS by a legitimate participant

Resilience to guarantee availability of the IT-EWS even under attack scenarios

Interoperability to guarantee compatibility to the existing infrastructure within the operator network (ON) and to enable collaboration with other Early Warning Systems and (external) sensors

Fairness to guarantee equal balance of input to and information received from the IT-EWS to maximize the return of investment for participating organizations

The relevance of each requirement may vary depending on the different types of collaborators participating in an IT-EWS and the level of trust they share between each other. Our CREW concept, however, provides a framework to fulfill all requirements even for IT-EWSs with a weak level of trust.

3 A Concept for Privacy-Preserving and Anonymous Information Sharing

In a common setting a network operator runs sensors within its local network, e.g., on network elements or user end devices, to detect attacks and anomalous behavior. These sensors transfer their collected – and sometimes already aggregated – data to a local central database inside the ON where it is analyzed and evaluated. Results of the analysis are stored locally and can be used to take countermeasures

against attacks to mitigate or even avert their negative impacts on the operator's infrastructure.

In some cases, however, the local data and results may not be sufficient to detect or counter an attack. Hence enriching the local network view by data from other administrative domains is advantageous. Therefore the collaborative IT-EWS enables network operators to share locally collected data with other participants, thereby providing information to them and consuming data provided by others.

We especially focus on the usage of existing components to address arising issues of collaboration and present how they can be combined to build an Information Sharing Network (ISN) for distribution of warnings and collaborative attack detection and analysis.

3.1 Components for Implementation

Although some of the requirements discussed in chapter 2 seem to be conflicting (e.g., anonymity and non-repudiation) we have identified specific components to fulfill all requirements especially for IT-EWSs with a weak level of trust between participants:

- Traceable Anonymous Certificates
- Anonymous P2P Overlay Networks
- Secure Multiparty Computation
- Common data exchange formats

These components are discussed in more detail in the following sections and Table 1 depicts the context between requirements and components.

Traceable Anonymous Certificates (TACs) [8] provide X.509 certificates containing pseudonyms instead of real identities in the subject line. A strict split of the Registration Authority (RA) and Certificate Authority (CA), a threshold signature scheme, and blinded signatures ensure that one authority alone is not able to link a TAC to a concrete user. However, in case of abuse of a TAC an aggrieved party can consecutively collaborate with the CA and RA to reveal the real identity of a TAC user. In CREW TACs can be used to provide authenticity, integrity, confidentiality, non-repudiation and anonymity.

Anonymous P2P Overlay Networks GNUnet[3] is a framework for secure P2P networking. Together with gap [15] it provides an anonymous overlay network allowing to distribute content in a P2P network without revealing the originator of this content. The usage of GNUnet and gap in the context of CREW is described in section 4.1.

[3] `https://gnunet.org/`

Secure Multiparty Computation The overall goal of Secure Multiparty Computation (MPC) is to enable joint computations on private input data from multiple participating parties without revealing the actual individual input data. The result of the computation, however, can be provided to all participants. Thus, MPC allows contributing private input data without the risk of losing reputation due to leakage of sensitive information included in the input data. This is especially interesting in the domain of collaborative incident handling and attack detection across multiple administrative domains. Since its initial presentation by Yao [16] in 1982 a variety of different protocols for a number of different application fields ranging from sugar beet auctions [9] to secure supply chain management [2] has been developed for MPC. The two most popular types of MPC are Secret Sharing and Homomorphic Encryption.

While homomorphic encryption uses encryption algorithms to obfuscate the input values of the participants, secret sharing is based on splitting up a participant's private input data into shares and distributing them to the other parties. In 2010 Burkhart et al. [7] presented an MPC library based on Shamir's Secret Sharing called SEPIA[4] (SEcurity through Private Information Aggregation) enabling the aggregation of network and security data between multiple administrative domains. This Java library supports a number of different protocols to process different types of input data depending on the desired type of result including Event Correlation, Network Traffic Statistics, and Top-k Queries.

By relaxing the constant round requirement Burkhart et al. were able to optimize performance of the protocols thereby making feasible near real-time analysis of network and security data.

Thus, MPC implementations such as SEPIA provide the flexibility and privacy-preserving techniques required for CREW and are therefore potential building blocks for collaborative anomaly and attack detection and analysis as outlined in section 4.2.

Data Format A data format, appropriate for sharing IT-security relevant information is another important building block of CREW. The sharing of warnings (see section 4.1) should enable participants who have detected malicious or at least suspicious behavior in their own networks to inform the other members of the ISN about the observed phenomenon. If possible the format should be capable to transport information – if applicable and available – about the exploited vulnerability, the attack steps, the issuer's estimation about the impact of the incident as well as potential countermeasures. A suitable data format ideally allows integration or collaboration with existing frameworks that handle IT-security relevant information.

[4] `http://sepia.ee.ethz.ch/`

A plethora of different data exchange formats for the delivery of warnings or the transmission of raw data has already been defined, each of which focusing on a particular setting or use case and thus differing from each other. Relevant examples are the Incident Object Description Exchange Format (IODEF)[10], the Intrusion Detection Message Exchange Format (IDMEF) [3], Real-time Inter-network Defense (RID) [12], the European Information Security Promotion Programme advisory format (EISPP) [14], the German Advisory Format (DAF) [1], and the Cybersecurity Information Exchange Techniques (CYBEX) initiative [5] [5].

When selecting candidates for use in CREW aliveness of the data format as well as the opportunity to exchange impact information should be taken into account. Furthermore the provisioning of advisories, recommendations and information about vulnerabilities should be possible. The sending of raw data such as Malware samples and the submission of information about attack steps are further requirements. Last but not least the interoperability with existing solutions should be considered. Of the above mentioned formats, IODEF and CYBEX seem to fit best.

3.2 Summary of Requirements and Components

Table 1 summarizes which requirement of chapter 2 is fulfilled by the above described components. Section 4 describes how these components are used to implement the two most relevant applications for an IT-EWS.

Table 1. Requirements for collaboration and components to fulfill them.

	TAC	P2P	MPC	Data Format
Anonymity	x	x		
Privacy			x	
Non-repudiation	x			
Interoperability				x
Resilience		x		
Authentication	x			
Integrity	x			
Confidentiality	x			
Fairness		x	x	

4 Applications for Collaborative Information Sharing

Brunner et al. [4] discuss the usage of TACs and P2P Overlay Networks to achieve anonymous and privacy-preserving information sharing for IT-EWSs. They focus on

[5] http://www.itu.int/en/ITU-T/studygroups/com17/Pages/cybex.aspx

a high-level concept for information sharing, i.e., on the combination and properties of the two mentioned components. [4] also mentions the usage of MPC for IT-EW as a future research activity.

CREW integrates the combination of the three proposed components TACs, GNUnet and MPC extended by a common data exchange format and develops protocols for the two most important aspects in an ISN such as CREW

- Sharing of warnings between participants
- Collaborative detection and analysis of attacks and anomalies

These two applications are described in more detail in the following sections.

4.1 Sharing of Warnings

When network operators detect attacks on their networks they are supposed to react to these incidents to minimize the negative effects on their infrastructure and to mitigate loss of money or reputation due to downtime of services, leakage of sensitive customer data etc.

If more than one network operator might be affected by a specific type of attack it may be advantageous for the operators to share information on the attack using CREW. The sharing of information can be done by distributing attack information in the terms of warnings by broadcasting them to the IT-EWS.

This leads to a considerable advantage for the participants not yet affected by an attack or not able to detect it and develop countermeasures on their own. If the warning is not relevant for a participant the received information may be stored for future use or just be discarded. Prerequisite for an effective and efficient sharing of warnings is a common data format such as IDMEF, IODEF or the like. This facilitates automated analysis and processing of the warnings and thus minimizes the time needed to act based on the received attack information and mitigate an attack by implementing appropriate countermeasures. For the distribution of warnings via the IT-EWS we recall the requirements and the components to fulfill them as shown in Table 1.

As base for CREW we create a closed user group using a full-meshed Virtual Private Network (VPN) between the participants based on X.509 certificates for authentication. The VPN in turn assures the integrity and confidentiality of the exchanged information while traveling through the IT-EWS. Building up a VPN based on authentication via certificates and signing the warnings using a certificate's private key also assures that no illegitimate entities such as attackers are able to join the IT-EWS and inject falsified warnings into the IT-EWS or intercept and manipulate legitimate warnings.

However, participants may not want to be identified as the originator of a warning since they fear loss of reputation when revealing that their network is attacked. Thus, for signing warnings, we need certificates providing authenticity, non-

repudiation and seemingly conflicting anonymity at the same time. This conflict, however, can be solved by revealing a participant's real identify in case of abuse. TACs as discussed in section 3.1 fulfill these requirements and allow to trace the real identity of a participant, e.g., when purposely injecting false information into an IT-EWS.

The TACs are issued by a CA trusted by all participants of an IT-EWS and can therefore be used for authentication like regular X.509 certificates. Although TACs provide anonymity for the certificate owner the real identity of a TAC owner may be inferred over time due to the type and content of warnings distributed to the IT-EWS. Hence all participants of the IT-EWS shall be equipped with multiple TACs or the TACs of all participants shall be replaced with new ones on a regular basis or after specific events such as a new participant joining the IT-EWS.

Although replacing the participants' TACs helps to mitigate the risk of inference attacks the real identity of the originator of a warning may be revealed due to its source IP address. Thus, CREW should implement another layer of anonymization such as Onion Routing or anonymous P2P Overlay Networks. GNUnet and its gap provide such an anonymous P2P network by indirection of requests and responses over multiple nodes in the network. Hence receiving a request or response from a specific node in the network does not mean that this node is the originator of the message in question.

Another source of information to reveal a participant's real identity is the content of the warning itself which has to be sanitized carefully before sent to the IT-EWS. Since the content and therefore potential sensitive data depend on the type of attack, sanitizing a warning is a nontrivial task which is hard to automate and should therefore be done manually by an expert. A warning, however, should at least contain the following data (if available):

- Type of attack
- Target of attack (rather descriptive, e.g., OS, software version etc.)
- Impact
- Solution/countermeasures
- Related known vulnerabilities (e.g. CVE numbers etc.)
- Further sources of information

This information should be well structured and put into a common data format agreed on by all participants of the IT-EWS. Suitable candidates for this data format are discussed in section 3.1.

Participants can either request warnings from CREW by querying the network in regular intervals or warnings are broadcasted to the IT-EWS as soon as they are available at one operator. Figure 1(a) depicts the latter process.

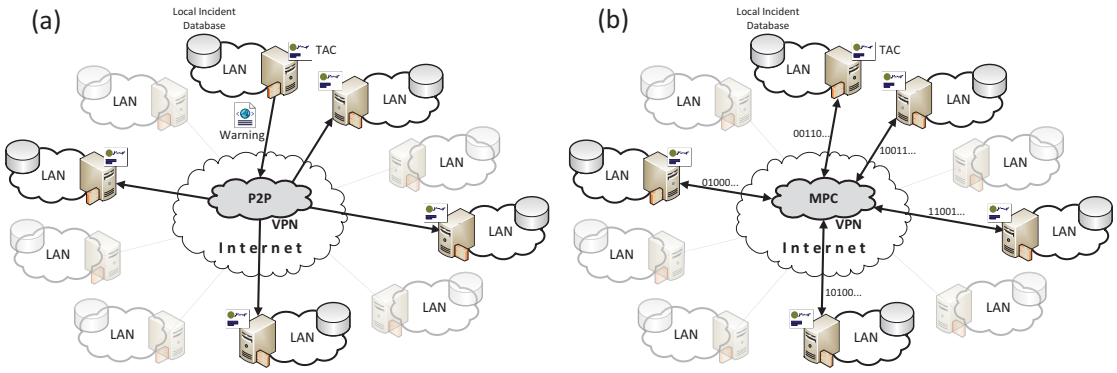


Fig. 1. (a) An anonymous P2P Network on top of a full-meshed VPN to distribute warnings within a closed user group. (b) The users exchange their MPC shares and intermediary results over secured communication channels.

4.2 Collaborative Detection and Analysis

Aggregating (sensor) data and attack information from multiple administrative domains using MPC techniques provides a bigger data set than from one single source.

The information required for collaborative detection and analysis differs from that used for sharing of warnings. While warning contents are already filtered either manually or automatically, information for MPC based processing should be in raw sensor data format. Hence the common data format for information sharing mentioned in section 4.1 is not suitable for the task discussed here.

Although MPC enables privacy-preserving data aggregation and sharing it does not make obsolete selection of the information to be shared. For practical collaborative detection and analysis the participants providing their private input data need to agree on the type of data to be analyzed and which parameters need to be shared. Reducing the parameters to be contributed to the collaborative detection and analysis, however, has rather performance implications than minimizing privacy issues.

Agreeing on the required parameters may be done using the warning application discussed in section 4.1. Instead of broadcasting a concrete warning via CREW this mechanism could also be used to request and initiate a new collaborative detection and analysis process including the configuration required for the appropriate MPC joint computation.

Potential types of data for collaborative detection and analysis may include but are not limited to:

- Malicious/suspicious IP addresses (IPv4 and IPv6)
- Scanned ports
- Netflows

- (Hashes of) Malware binaries
- Malicious/suspicious URLs (e.g. for dropzones)
- Properties of attacked devices
- Attack timestamps

Although secret sharing based protocols provide privacy and fairness to CREW participants, MPC normally does not fulfill other requirements as listed in chapter 2. Thus, SEPIA provides an integrated P2P network based on SSL connections thereby fulfilling the requirements resilience, authentication, integrity and confidentiality. The required certificates to enable SSL connections need to be created beforehand and added to the configuration of the SEPIA peers. Figure 1(b) depicts the exchange of data shares for MPC joint computations between participants of a CREW-based IT-EWS.

5 Conclusion

In this paper we describe a concept for secure and privacy-preserving information sharing between administrative domains. Our CREW concept fulfills all important requirements of an IT-EWS regarding loss of reputation and privacy considerations and helps to encourage information exchange even between competitors. Our approach uses existing components like Traceable Anonymous Certificates, Anonymous P2P Overlay Networks, Secure Multiparty Computation, and a common data exchange format. We also depict how these components can be combined to build the two most important applications for collaborative information exchange, namely sharing of warnings and collaborative detection and analysis thereby protecting the participants' interests security, data privacy, and originator anonymity.

References

1. D. C. Verbund. German Advisory Format (DAF). http://www.cert-verbund.de/daf/daf_description.html.
2. F. Kerschbaum, A. Schroepfer, A. Zilli, R. Pibernik, O. Catrina, S. de Hoogh, B. Schoenmakers, S. Cimato, and E. Damiani. Secure collaborative supply-chain management. *Computer*, 44(9):38 –43, sept. 2011.
3. K. Moriarty. Real-time Inter-network Defense (RID), Nov. 2010.
4. M. Brunner, H. Hofinger, C. Roblee, P. Schoo, and S. Todt. Anonymity and privacy in distributed early warning systems. In *CRITIS 2010: Proceedings of the 5th International Conference on Critical Information Infrastructures Security*, pages 82–93. LNCS, 2010.
5. A. Rutkowski, Y. Kadobayashi, I. Furey, D. Rajnovic, R. Martin, T. Takahashi, C. Schultz, G. Reid, G. Schudel, M. Hird, and S. Adegbite. CYBEX: the cybersecurity information exchange framework (x.1500). *Computer Communication Review*, 40(5):59–64, 2010.
6. M. Apel, J. Biskup, U. Flegel, and M. Meier. Early Warning System on a National Level - (Project AMSEL). In *Proceedings of the International Workshop on Internet Early Warning and Network Intelligence (EWNI)*, 2010.

7. M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. In *USENIX Security Symposium*, pages 223–240, 2010.
8. S. Park, H. Park, Y. Won, J. Lee, and S. Kent. Traceable Anonymous Certificate. RFC 5636 (Experimental), Aug. 2009.
9. P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *Financial Cryptography*, pages 325–343, 2009.
10. R. Danyliw, J. Meijer, and Y. Demchenko. The Incident Object Description Exchange Format (IODEF). RFC 5070 (Proposed Standard), Dec. 2007.
11. S. Pinkerton. A Federated Model For Cyber Security. In *Cyberspace Research Workshop, Shreveport, LA*, November 2007.
12. H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental), Mar. 2007.
13. K. Kossakowski, J. Sander, B. Grobauer, and J. I. Mehlau. A German Early Warning Information System - Challenges and Approaches. Presentation at 18^{th} Annual FIRST Conference, June 2006.
14. EISPP-Consortium. EISPP Common Advisory Format Description, Identifier: EISPP-D3-001-TR, v2.0. Technical report, 2004.
15. K. Bennett and C. Grothoff. gap - Practical Anonymous Networking. In *Privacy enhancing technologies. International workshop No3, Dresden*, volume 2760 of *Lecture notes in computer science*, pages 141–160, 2003.
16. A. Yao. Protocols for secure computations. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:160–164, 1982.

IO: Deploying An Interconnected Asset Ontology To Enhance Information Retrieval Regarding Security Processes

Henk Birkholz

Universität Bremen TZI, Germany
birkholz@tzi.de
http://www.tzi.de

Abstract. Security-related processes require well structured asset information. The IO framework provides interconnected asset information in a manner that enables its flexible utilization. This paper presents advances in the development of IO: A terminology is proposed that adapts definitions found in the context of organizational memory information systems. A generic concept of "Group" enables categorizing assets, enhancing information retrieval. New use cases demonstrate the interaction of IO with producers and consumers of information in the context of information retrieval.

1 Introduction

In a production environment security-related decision processes depend on detailed asset information [15,24,16]. The interconnected-asset ontology (IO) framework satisfies this demand and provides a structured representation of detailed asset information [11]. However, introducing a framework, such as IO, in an organization is no simple endeavor. Without an urgent demand or a direct incentive [6], deployment of an IT solution usually is a proactive task. In general, direct benefits make the corresponding investment of time and resources more attractive to stakeholders. Hence, we present in this paper characteristics and applications of IO that provide direct incentives to encourage its proactive deployment in an organization: A group concept taxonomy enhances and simplifies information retrieval to support heterogeneous security processes. In addition, the applications presented in this paper can provide further incentives promoting proactive deployment of the IO framework: the automatic generation of Nagios configuration [8] and the support of a deployment process introducing network domain security (NDS) design principles into existing mobile communication IT-infrastructure specified by the 3rd Generation Partnership Project (3GPP) [2]. An additional contribution of this paper is to introduce a terminology adapted from organizational memory information systems (OMIS). We show, that in a top level view, the primary functions of the IO framework are very similar to mnemonic functions found in OMIS.

The remainder of this paper is structured as follows: In section 2 we discuss research on organizational behavior, proactivity and motivation as relevant factors

promoting a successful deployment of the IO framework. Related work on asset categorization is presented to emphasize the usefulness of a group concept regarding the information retrieval process. The related work section also covers organizational memory information systems (OMIS): Similarities between basic functions implemented in the IO framework and mnemonic functions found in OMIS are highlighted. Section 3 describes the design of IO with regard to the function terminology adopted from OMIS. Characteristics of information producers and information consumers in the context of asset information are also included in section 3. In section 4 the group concept is introduced and examples of its application are presented. In section 5 the use cases and their corresponding scenarios are described. The soundness of the approach is illustrated via pseudo code for each use case.

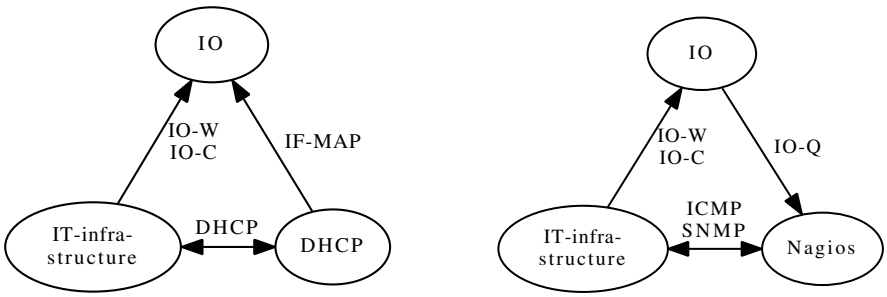
2 Related Work

Asset Categorization Ault et. al. highlight asset management as an important prerequisite for short and long term decision processes [7]. On top of that, the association of assets in groups and categories is a common procedure that further enhances decision processes. This is discussed, for example, by A. Rajakrom in the area of knowledge engineering [30] or by Likar and Trcek with application in sustainable information security [24]. Aggregating assets in a set and then processing the resulting group of assets as a single entity provide certain benefits: Annotating context information is resource intensive. By reducing the number of entities, the resources required to annotate assets can be significantly reduced, with no loss in quality regarding the annotated features. Information retrieval becomes easier, because more general characteristics associated with a group of assets can be used as an identifier. This is especially useful if more specific or more precise identifiers to initiate an information retrieval are rare.

Asset categorization is also found as a common step in information security (IS) methodologies, such as the German IT-Grundschutz [12], EBIOS [33], or Octave [5]. In these examples, the desired goal is reducing complexity in further manual – or automated – process steps and thereby reducing cost. The use of asset categories can also be found in availability monitoring frameworks like Nagios[1] or Zabbix[2]. In these frameworks, network related IT assets are often grouped by the use of templates, defined by similar asset properties: e.g., location, responsible contact, type of asset or type of provided service. This reduces redundancy in configuration and therefore the amount of resources needed to keep configuration aligned with existing IT infrastructure. Clustering of assets in distinct sets that possess similar features or requirements is an integral part of various security related specifications, such as the technical specification of Network Domain Security (NDS) for IP Based

[1] http://www.nagios.org/
[2] http://www.zabbix.com/

Protocols defined by the 3G Partnership Project [2]. In this specification, a security domain is composed of IT assets of one authority that share the same level of security and usage of security services. Essentially, defining a security domain is a decision process regarding IT asset categorization.



(a) Producer of Information (b) Consumer of Information

Fig. 1. Examples for Producer and Consumer of Information

Organizational memory information systems Abecker and Decker describe an Organizational Memory Information System (OMIS) as "basic techniques integrated into a computer system which within the enterprises' business activities continuously gathers, actualizes, and structures knowledge and information". OMIS provides knowledge in different operative tasks in a context-sensitive, purposeful and active manner in order to improve cooperative, knowledge-intensive work processes [3]. The IO framework adopts several of these principles – as do other knowledge base centered approaches [42,10] – and applies them to ontologically structured IT asset information. An OMIS incorporates several functions based on principles of information storage and retrieval in society [23]. These principles were applied to an IT-supported Information System by Stein and Zwass [35]. They focus on updating central repositories, containing basic and context information about an organization, and specify mnemonic functions as shown in figure 3. The requirements presented in [35] regarding Knowledge Retention (e.g. structures for different encoding, or models for organizing past and present knowledge) are satisfied by an ontology represented in OWL [39]. Hence, IO uses similar functions which will be presented in section 4.

Organizational behavior and proactivity In order to retrieve information from an ontology in an ad-hoc fashion (with as little latency as possible) it is mandatory to acquire the necessary asset information out of the current IT infrastructure proactively. This task can require an investment of technical and organizational resources without directly visible benefit. This kind of proactive behavior is important [22],

but it is also difficult to achieve. Potential reasons for this (e.g. incentives or influencing factors called enabler and disabler) are discussed in the field of organizational behavior: research strongly supports that the availability of security related resources improves the quality and performance of security related actions [20]. This can be an indirect incentive for stakeholders to deploy IO in an organization.

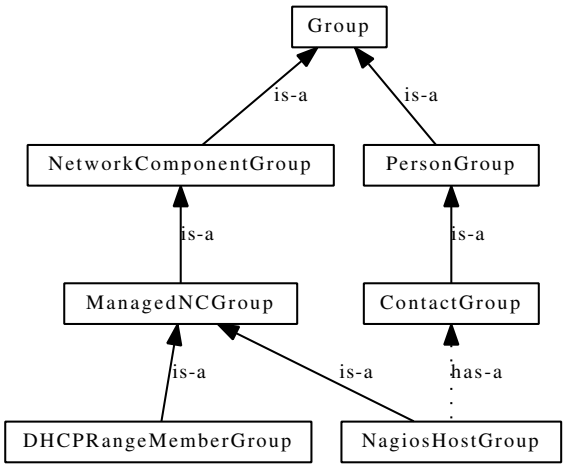


Fig. 2. Excerpt of group concept taxonomy

Introducing direct incentives in an organization to enhance proactive behavior and overall information security is a complex and ceaseless process [17]. Understanding some of the underlying principles can help increasing its effectiveness: A key element in organizational behavior research is understanding and measuring motivation, which is therefore an important factor to enable proactive deployment of IO. There are various definitions of motivation throughout the literature of the last 80 years. One of the best known theories of motivation is proposed by Maslov [25]: He speaks of a hierarchy of driving forces (needs); the most idealized one, for example, being self-actualization – the motivation to "become". In an organizational sense, motivation has been described as "the set of processes that arouse, direct, and maintain human behavior toward attaining a goal" [19]. Parker et al. provide a simpler example by defining the intensity of motivation in the traditional meaning of organizational behavior: "how much effort one is prepared to put in [a task]" [28]. Hence, supporting motivation by, e.g. increasing security awareness [21] or well placed incentives [6], proactive deployment of IO becomes more attractive. Research of motivation regarding security processes has placed a strong emphasis on better promoting and enabling security policies [34,38,32]. In these findings, the online availability of security policies and further context information has a significant positive impact on an individual's motivation to be compliant with the policies:

Information which is available online reduces the amount of effort necessary to be compliant. Making information available through information technology is also an enabler for better understanding and using complex knowledge [31]. This again is an indirect incentive for stakeholders arguing towards the deployment of the IO framework. If an acute threat has to be assessed, or if a security requirement in a project is suddenly deemed mandatory in its late stages [13], the availability of context information about the IT infrastructure increases the quality of the corresponding decision process and therefore its outcome.

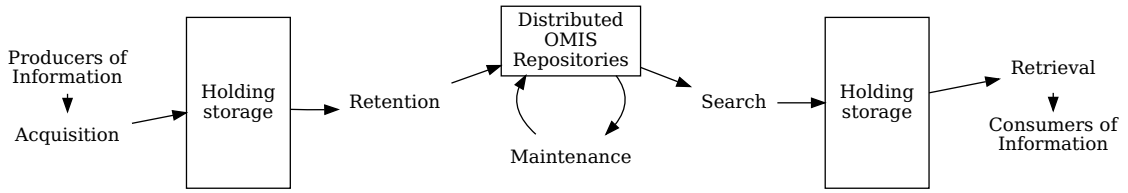


Fig. 3. Mnemonic Functions of an OMIS [35]

In contrast to the definition of Parker et al., the waste of effort is a well known disabler regarding motivation [41]. Effort wasted on redundant activities can have a deleterious effect on task performance. For example, redundant activities can result in violation of time and resource limits, fatigue, poor learning, poor task knowledge or even uncompleted tasks due to quitting. As mentioned above, Stein and Zwass argue that making information available with information technology can be broken further down into subprocesses like acquisition, maintenance or retrieval [35]. Redundant activities (as a waste of effort) can accumulate over these subprocesses as part of making information resources available, amplifying this disabler. The IO framework focuses on eliminating redundant acquisition processes, centralizing these tasks in a way that information about specific categories of IT assets (e.g. managed network components) has only to be acquired once and is then available in a well-structured manner to support any further security related process.

3 Components of the IO framework

In order to identify effective incentives to advocate proactive deployment of IO we highlight each major function of IO and the potential benefits they can provide regarding security-related processes. Adapting the basic principles found in Organizational Memory Information Systems, the IO framework is divided into modules according to OMIS functions: acquisition, retention, search and retrieval. Figure 4 shows IO functions represented by its modules. Asset information is produced by or gained form IT infrastructure. Some of it is manually produced by the employees of

an organization (e.g. static configuration) and some of it is generated automatically (e.g. layer 2 address associations on a switch port). Information is consumed by any task-specific security process that can be supported or enhanced by IT asset information. In this section basic OMIS functions and their implementation are discussed. Correspondingly, examples of producers and consumers of information are described and their characteristics are generally categorized.

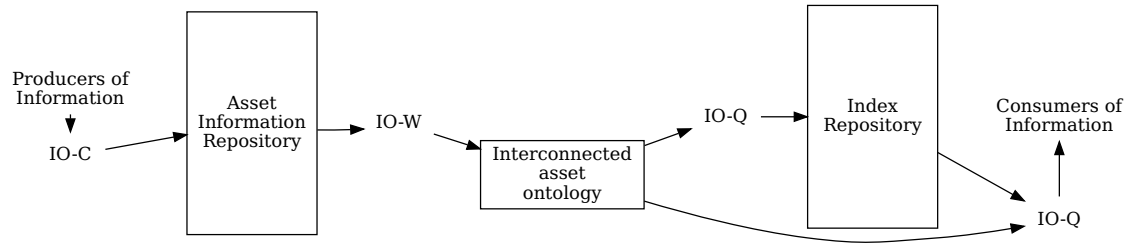


Fig. 4. Adaption of function terminology in IO

Producers of information Producer of asset information generally provide two major categorizes of information: static information and dynamic. Categorization of producers of information by the type of information produced is necessary because the categorization influences corresponding information acquisition procedures. There are properties by which both groups of information can be discerned: Dynamic information, for example, often is generated by the IT systems themselves. While this dynamic generation might be based on static configuration, e.g. a scope of layer 3 addresses available for dynamic distribution, the actual state of dynamic information must be extracted from the IT infrastructure in order to be known precisely. This can be achieved by gathering information either from the providing end or the receiving end of dynamic information (e.g. DHCP server or DHCP client). While it is more efficient to acquire information from the providing end – simply because fewer acquisition procedures are required – dynamic information regarding an asset is then only known implicitly. Higher information integrity is achieved if the IT asset utilizing the asset information is directly involved in the acquisition process, ruling out errors in the distribution process of dynamic information between IT assets.

The mechanism used to acquire IT asset information is another indicator useful to discern between static and dynamic information. Dynamic information is prone to high fluctuation. Therefore it has to be updated via IO more often than static information. This makes a push mechanism far more desirable than a poll mechanism. If an IT asset contains a service that can push information about its state only when its state really changes, the overall resources needed to acquire dynamic information are reduced significantly. Using poll mechanisms, the IO framework has

Listing 1: Build Nagios network map

```
1: root ← get_root_device_for_nagios_map
2: devices ← get_nagios_monitored_devices
3: for d in devices do
4:   path ← get_path(root, d)
5:   parent ← get_next_to_last(path)
6:   write_nagios_host_entry(d, parent)
7: end
```

to acquire the whole set of information provided by an IT asset and then compare it with the last known state.

Hardware identifiers and software configuration located directly on an IT asset for internal use are the most common example for static information acquired from producers of information. There also exists dynamic information that is, regarding its fluctuation, almost similar to static configuration: neighborhood relationships between core and distribution network components. Neighborhood relationships dynamically discovered between managed network components (via the use of, e.g. link layer discovery protocols) rarely change. In contrast, neighborhood relationships with mobile devices are very short lived: A wireless client unsuccessfully trying to connect with an access point repeatedly can result in hundreds of neighborhood relationships within seconds. These highly volatile relationships cannot be processed by the IO framework efficiently. In this domain, however, other solutions apply: The IF-MAP specification [37], originating in the context of the TNC specification, is on one hand able to handle a higher fluctuation rate, but on the other hand lacks the features to represent more static relationships as is the objective of IO. Figure 1(a) shows an example of a producer of information pushing asset information on layer 3 address changes via IF-MAP to IO.

Acquisition of information The IO-Collector (IO-C) module is responsible for the acquisition process in the IO framework. Its goal is to extract raw information about configuration and asset state out if the IT infrastructure. Raw asset information is stored in an asset information repository. It handles temporary loss of asset availability during acquisition procedures and provides an extensive logging engine to isolate root causes if no information can be acquired about an IT asset. It is important to differentiate between permanently removed or temporary unavailable IT assets in order to reduce number and size of updates. Protocols used in the acquisition process primarily are: SSH, SOAP, SNMP, LDAP and IF-MAP. Information can be acquired via the IO-Collector in intervals from one minute to days or weeks. Too small intervals can increase the load on IT assets unnecessarily. Hence, the definition of e.g. polling intervals has to be aligned with fluctuation of data and the computing resources available.

Listing 2: Group monitored devices by building

```
hostgroups ← new(Hash)
devices ← get_nagios_monitored_devices
for d in devices do
  building ← get_building_from_name(d)
  if !hostgroups.include?(building) then
    hostgroups[building] ← new(Array)
  end
  hostgroups[building].append(d)
end
for building, devs in hostgroups do
  write_nagios_hostgroup_entry(building)
  for d in devs do
    add_member_to_hostgroup(building, d)
  end
end
```

Retention of information Acquired interconnected asset information is stored in an ontology via the IO-Writer (IO-W) module. Depending on source, encoding and syntax a modular set of parsers is used to extract individuals, data properties and object properties according to the ontological concept layout used by IO (represented in OWL). Examples are: a neighborhood relationship is represented by an object property chain. Discernible physical or virtual IT assets are each represented by an individual. An interface status (e.g. up, down, administratively down or error disabled) is represented by a data property. In some cases it is difficult to discern a virtual IT asset from a physical IT asset based on acquired raw information alone. If asset information about a hosting IT system contains meta data about its hosted virtual IT systems, it is possible to derive the appropriate object properties automatically during the retention process. In other cases where, e.g., high availability of core routers is achieved by seamless virtualization over different physical devices, predefined and device specific context knowledge is necessary and included in the parser modules to enable correct classification and representation in IO.

Listing 3: Generate clusters by security requirements

```
devices ← get_devices
clusters ← new(Clusters)
for d in devices do
  neighbors ← get_neighbors(d)
  security_requirements ← get_security_requirements(d)
  for n in neighbors do
    n_security_requirements ← get_security_requirements(n)
    if n_security_requirements = security_requirements then
      clusters.add_cluster_membership(d, n)
    end
  end
end
```

Search and retrieval of information Search and retrieval procedures are both conducted by the IO-Query (IO-Q) module. The search function offers a SPARQL [29] interface in order to infer information via the use of a reasoner. Reoccurring search operations initiated by several retrieval functions are cached as indexed result lists (IRL) in an index repository. Some essential IRLs are automatically generated after an ontology update. Time-critical retrieval operations, as can be found in the SIEM context, can make deliberate use of automatically generated IRLs during information retrieval and thereby increase retrieval performance significantly [11]. Retrieval procedures can also make direct use of the search function. Output of the retrieval function is primarily composed of: REST, JSON, CSV or SCAP-AI format. Further application specific output can be generated via optional output modules.

Consumer of information Interconnected asset information is useful in various security related processes [9,18]. Figure 1(b) shows an example introducing a consumer (a Nagios network monitoring service) of information utilizing the interconnected asset information provided by IO. In this example it is important to keep the ontology updated frequently. This can be supported by the example of a producer of information 1(a): IP address changes of monitored IT assets should be pushed in real-time to update IO. The following sections introduce a group concept and focus on requirements regarding consumer of information.

4 Group concept in IO

In an ontological representation an asset categorization can be expressed in two ways: by the use of data properties or by the use of object properties. While the use of data properties doesn't increase the complexity of the concept layout, it increases response times of the search function: every type of individual that could potentially be a member of a group has to be evaluated by the reasoner. Using object properties to associate members of a group with a corresponding group individual, a reasoner can select appropriate individuals more efficiently while conducting a search operation. Figure 2 shows the group concept layout in IO, which is structured taxonomically. Examples of group individuals representing categorization of assets are: "internal layer 3 addresses" or "external layer 3 addresses", "located in building" or "located in administration building", "IT-Grundschutz module Linux server" [12] or "Nagios http server object". Application specific categorizations, such as Nagios templates, are available in IO and can be used by other consumers of information if the need arises. Especially intelligent algorithms, as used in the FIDES framework [1], benefit from detailed annotated context information, supporting the detection of new, hidden dependencies in event streams.

Listing 4: Validation of SD requirements

```
1: firstSD ← get_sd_members(first)
2: secondSD ← get_sd_members(second)
3: invalid ← new(Array)
4: for f in firstSD do
5:   for s in secondSD do
6:     path ← get_path(f, s)
7:     if!contains_sg(path) then
8:       invalid.append([f, s])
9:     end
10:  end
11: end
12: return invalid.empty?()
```

5 Use Cases

Automatic generation of Nagios configuration We present two use cases regarding the automatic generation of application specific configuration from interconnected asset information: First we show how to generate a Nagios configuration without any previously annotated categorization in use case 1a presented in listing 1. In use case 1b (listing 2) we demonstrate how to automatically annotate group membership by defining hostgroup templates in a Nagios configuration.

Automatic support of network domain security according to the 3GG NDS specification The 3GG network domain security specification defines security domains (SD). Security domains are groups of interconnected assets in a network sharing identical security requirements. One group of assets composing a SD is only allowed to communicate with another SD via security gateways (SG). SGs therefore should provide the only possible communication paths between SDs. Use case 2a, presented in listing 3, shows how to group assets into a potential SD according to their security requirements. Use case 2b validates a given categorization of SDs (represented by a group of assets), as presented in listing 4, by identifying potential invalid communication paths between SDs that do not traverse a SG.

shows how to group assets into a potential SD according to their security requirements. Use case 2b validates a given categorization of SDs by identifying invalid communication paths between SDs, not traversing a SG and violating security requirements.

6 Conclusion

A primary objective of this paper is introducing a terminology that originaly came from OMIS and adapting it to the workflow of the IO framework. A complete OMIS implementation includes a significant larger scope of context or meta information than IO does. This complexity made the deployment of an OMIS challenging. In

comparison, IO's specific approach – concentrating on asset-related context information – results in a far less complex concept layout. In combination with its generic group concept, this is enables the acquisition of more complete context information which eases the utilization of the IO framework in practice.

But the initial effort to deploy IO in an organization can still be high: automatic acquisition procedures must be implemented and accessing sensitive infrastructure information always presents a significant threshold that has to be overcome. It is important to provide direct benefits as an incentive to deploy a framework such as IO. The use cases presented offer such a direct benefit and we can show that the results are plausible and directly usable. After a successful deployment, further security processes are enabled to access a proactively provided pool of context information, reducing redundant acquisition processes and thereby cost. To provide detailed asset and context information to virtually any consumer of information a scalable mechanism enabling search functions is necessary: The IO group concept taxonomy – as a step towards a standardized representation of detailed asset information – satisfies this requirement and enables information retrieval even if identifiers are sparse.

References

1. FIDeS development website. `http://www.fides-security.org`
2. 3rd Generation Partnership Project. 3G Security; Network Domain Security (NDS) – IP Network Layer Security, 2011.
3. A. Abecker and S. Decker. Organizational memory: Knowledge acquisition, integration, and retrieval issues. In *Knowledge-Based Systems. Survey and Future Directions*, volume 1570 of *LNCS*, pages 113–124. Springer, 1999.
4. A. Abecker et al. Organizational memory. *Informatik-Spektrum*, 21:213–214, 1998.
5. C. J. Alberts and A. J. Dorofee. Managing information security risks the OCTAVE approach, 2003.
6. R. Anderson. Information security economics - and beyond. In *Deontic Logic in Computer Science*, volume 5076 of *LNCS*, pages 49–49. Springer, 2008.
7. G. W. Ault et al. Asset management investment decision process. 2004.
8. W. Barth. *Nagios: System And Network Monitoring.* No Starch Press Series. Open Source Press, 2006.
9. L. Beaudoin and P. Eng. Asset valuation technique for network management and security. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 718 –721, 2006.
10. A. Birk and F. Kröschel. A knowledge management lifecycle for experience packages on software engineering technologies. In *Learning Software Organizations*, volume 1756 of *LNCS*, pages 142–160. Springer, 2000.
11. H. Birkholz et al. IO: An interconnected asset ontology in support of information security applications. In *7th International Conference, Availability, Reliability and Security*, Prague, Czech, 2012.
12. BSI. German IT Baseline Protection Manual. 2011.
13. P. T. Devanbu and S. Stubblebine. Software engineering for security: a roadmap. In *Proc. of the Conference on The Future of Software Engineering*, ICSE '00, pages 227–239. ACM, 2000.
14. P. T. Devanbu and S. Stubblebine. Software engineering for security: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 227–239, New York, NY, USA, 2000. ACM.

15. A. Ekelhart et al. Formal threat descriptions for enhancing governmental risk assessment. In *Proc. of the ICEGOV 2007*, pages 40–43, New York, NY, USA, 2007. ACM.
16. A. Ekelhart et al. Ontology-based decision support for information security risk management. In *Proc. of the 4th IEEE SMC 2011*, pages 80–85, Washington, DC, USA, 2009. IEEE Computer Society.
17. E. Gal-or and A. Ghose. The economic incentives for sharing security information. *Information Systems Research*, 16:186–208, 2005.
18. A. Garland et al. Characterization of network topology to support infrastructure asset management. *Public Works Management & Policy*, 14:81–101, 2009.
19. J. Greenberg and R. Baron. *Behavior in organizations: understanding and managing the human side of work*. International student edition. Allyn and Bacon, 1993.
20. T. Herath and H. R. Rao. Protection motivation and deterrence: a framework for security policy compliance in organisations. *EJIS*, 18(2):106–125, 2009.
21. D. Johnson and H. Koch. Computer security risks in the internet era: Are small business owners aware and proactive? In *Proc. of the HICSS 2006*, volume 6, page 130b, 2006.
22. A. C. Johnston and R. Hale. Improved security through information security governance. *Commun. ACM*, 52(1):126–129, 2009.
23. K. Krippendorff. Some principles of information storage and retrieval in society. *General Systems*, 20:15–35, 1975.
24. B. Likar and D. Trcek. A Methodology for Provision of Sustainable Information Systems Security. *Cybernetics and Systems*, 43(1):22–33, 2012.
25. A. Maslow. *Motivation and personality*. Harper's psychological series. Harper, 1954.
26. M. Maybury et al. Analysis and Detection of Malicious Insiders. Technical report, 2005.
27. N. Nagappan. *A software testing and reliability early warning (strew) metric suite*. PhD thesis, North Carolina State University, 2005.
28. S. K. Parker et al. Modeling the Antecedents of Proactive Behavior at Work. *Journal of Applied Psychology*, 91(3):636–652, 2006.
29. E. Prud'hommeaux and A. Seaborne. *SPARQL Query Language for RDF*. W3C recommendation, 2008.
30. A. Rajakrom et al. Asset Categorization for Enhanced Asset Management Using Object Oriented Approach. 2006.
31. D. Robey et al. Information technology and organizational learning: a review and assessment of research. *Accounting, Management and Information Technologies*, 10(2):125 – 155, 2000.
32. A. M. Saks and M. Belcourt. An investigation of training activities and transfer of training in organizations. *Human Resource Management*, 45(4):629–648, 2006.
33. Secrétariat général de la défense nationale. EBIOS Section 1 Introduction, 2004.
34. M. Siponen. A conceptual foundation for organizational information security awareness. *Information Management and Computer Security*, 8(1):31–41, 2000.
35. E. W. Stein and V. Zwass. Actualizing organizational memory with information systems. *Information Systems Research*, 6(2):85–117, 1995.
36. A. Syalim et al. Comparison of risk analysis methods: Mehari, magerit, nist800-30 and microsoft's security management guide. In *ARES 2009*, pages 726 –731, 2009.
37. TCG Trusted Network Connect. TNC IF-MAP Binding for SOAP, 2012. Version 2.1 Revision 15.
38. K. Thomson and R. von Solms. Information security awareness: educating your users effectively. *Journal of Management Information Systems*, 11(1):167–187, 1998.
39. W3C. OWL 2 Web Ontology Language Document Overview. Technical report, 2009.
40. A. Wagner et al. Experiences with worm propagation simulations. In *Proceedings of the 2003 ACM workshop on Rapid malcode*, WORM '03, pages 34–41, New York, NY, USA, 2003. ACM.
41. R. E. Wood et al. Motivation and information search on complex tasks. In *Work motivation in the context of a globalizing economy*, pages 27–48. 2001.
42. M. Yi. Information organization and retrieval using a topic maps-based ontology: Results of a task-based evaluation. *Journal of the American Society for Information Science and Technology*, 59(12):1898–1911, 2008.

Between Early Warning and Emergency Response - An economical perspective -

Heiko Kirsch[1] and Michael Hoche[2]

[1] Secure Mobile Networking Lab,
Technische Universität Darmstadt, D-64293 Darmstadt, Germany
heiko.kirsch@seemoo.tu-darmstadt.de
[2] Integrated Systems Engineering, EADS Deutschland GmbH / Cassidian
D-88039 Friedrichshafen, Germany
michael.hoche@cassidian.com

Abstract. This contribution outlines different perspectives on risk management applications for early warning and emergency response. The set of best practices and standards in the area of risk management is enriched by an economic perspective based on analytical foundations comprising concepts like machine learning, game theory or mechanism design borrowed from economic and computer science. We derive a cohesive approach to enable risk-informed decisions in the adverse security environment for recommending rational, adequate emergency response. Our primary focus is on decision support for optimal service value instead of focusing optimizing resource management. We lay down a concept to establish a mechanism that ensures proactive and reactive implementation of security countermeasures to meet real economical security needs.

1 Introduction

Due to our society's increasing demand of reliable and secure information technology and emerging regulatory, i.e. legal obligations to ensure resilience of infrastructures providing services, especially in terms of critical information infrastructures, there are needs for an economical approach to cope with inherent threats and vulnerabilities (see [19]). Despite the ongoing implementation of the requirements defined by the European Union (see [3]), the adoption of the "Communication on *Critical Information Infrastructure Protection (CIIP)*" (see [4]) for establishing "policies to strengthen the security of and the trust in information infrastructures" is still not fully accomplished. The further definition of concrete development plans on CIIP is based on the following five fundamental areas as defined in [4]:

1. **Preparedness and prevention**: Defining of baseline capabilities and services for emergency response to enable information sharing and best practice exchange (see [6] and [7])
2. **Detection and response**: Provisioning of adequate early warning mechanisms, reaching out to citizens and organizations to enforce public-private partnerships (see [10] and [8])

3. **Mitigation and recovery**: Reinforcing of comprehensive defense mechanisms, e.g. by national contingency plans and regular exercises, to establish trust relationsship (see [9])
4. **International cooperation**: Enabling closer pan-European coordination and strengthening international cooperation to align legal and regulartory requirements (see [5])
5. **Criteria for Development**: Identifying information infrastructure critical for the society and supporting future implementation of CIIP (see [12])

The formulated need for adequate preparation to cope with emergency situations regarding information security incidents at organizational, national, European and international level is facing to divergent, and sometimes competive interests of involved stakeholders. They have to handle a diversity of legal and regulatory frameworks (see [5]), an a priori opaque risk situation due to the asynchronous and asymmetric character of emerging threats (see [24]) and retain competitive advantages by efficient and effective service delivery to their customers (see [1]).

However, even if providers are aware of information security as enabler for continuous service provisioning following all available best practices provided by the *European Network and Information Security Agency (ENISA)*, the *International Organization for Standardization (ISO)*, the *National Institute of Standards and Technology (NIST)*, or the *German Federal Agency for Information Security (BSI)* to implement proactive measures to ensure information security due service operation, i.e. [13], [23] and [2], residual weaknesses are likely to remain (see [17]). This non-knowledge renders controls ineffective and thus information security incidents possible, potentially with both direct and indirect adverse impacts on an organization's business operations. Further, inevitably new previously unidentified or continuous evolving threats will occur. Adequate preparation by an organization in terms of efficient and effective management of such incidents and emergency situations needs an economical perspective to address emerging risks (see [11]).

2 Fundational Concepts

Classically, implementing an effective response capability involves several decision making and operational processes (see figure 1). This include definition of policies, objectives, and responsibilies within the organizational structure. Considerations about policies requires an organization-specific definition of terms like "informations security events" and "information security incident" including "early warning" and "emergency". Usually this perspective lacks the dimension of collaboration since it focusses to only one organization. The approach leads to an organizational boundary focussing on legal and regulatory requirements. As a consequence in network like systems — where many parties or organizations contribute — there is a lack of

securing emergent properties. Because definitions are slightly varying between the standard-developing organizations, we derive the follwing summary.

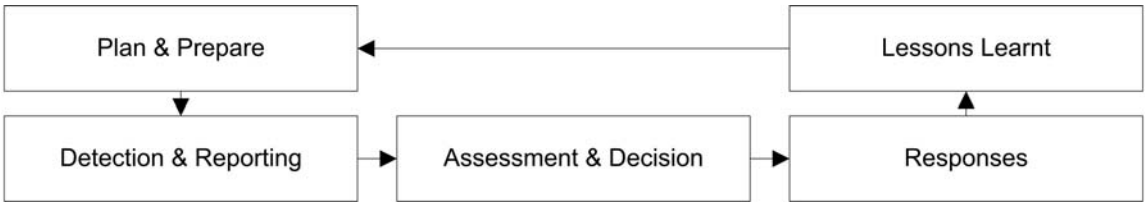


Fig. 1. "Information Security Incident Management" process based on [18]

2.1 Monitoring and Early Warning

The first challenging part of the incident response process is accurately detecting and assessing possible events, and determining reliable whether an incident has occurred. Thus the deployment of detection methods within the infrastructure is needed. This can be approached by continuous monitoring (see [23] or [13]), i.e. to collect information about the operational state of provisioned and consumed services and the underlying technical and organizational systems. Primary intend is to identify indications and precursors for emerging attacks and security incidents to establish proactive measures. The interpretation of events in terms of precursors can be considered as early warning, if identified. The information gained form a monitoring system can serve as input for further risk-driven prioritization for incident handling procedures as defined e.g. in [18] or [22].

2.2 Information Securty Events and Incidents

In [14] information security incidents are considered as adverse events. Information security events can be "captured" by deployed detection mechanisms sparsely or as a series of similar events. Whether the detection result is correct or not depends roughly on the recall and the precision of the detection mechanism. Security incidents have certain probability of compromising business operations and threatening information security in terms of e.g. confidentiality, integrity and availability.

2.3 Incident Handling and Emergency Response

Incident handling as ongoing process comprises any respond to information security incidents, including the activation of appropriate controls for the prevention and reduction of impacts of, and hence the recovery from service derivations (see [22]).

Thus appropriate incident handling enables timely reaction and continuous service delivery. As far as the estimated risk induced by incidents is at "severe" level also the terms "emergency" and "emergency response" are used (see [18]).

2.4 Risk-Informed Decision

A risk-informed decision is decision making, in which insights from probabilistic risk assessments are considered with other insights (see [20]). In mature organizations that manage complex insecure or unpredictable objects, risk-informed decision making is organizational embedded. Often there is a guidance for a procedure that comprises risk assessment, decision, mitigation and control cycle, i.e. [21] or [16]. The challenge within this approach is archiving the needed correctness and completeness of the assessment of assets and expected impacts from the point of view of all impacted stakeholders (see [1]).

To circumvent this difficulties in an adverse complex environment we propose a pragmatic collaborative approach measuring the real economical value of assets to be protected as well as the observed deficits, that are due to anomalies. To reduce the complexity introduced by misaligned strategies of incorporated parties we provide a generic assessment model incorporating the notions agency, strategy, interaction, behavior, value and deficit. This allows in advance to render economic and intentional perspectives for any stakeholder group.

3 Economic Perspective

We deviate from the conceptual approach of economics of information security that has recently become a thriving and fast-moving discipline. Instead we intend to identify advantageous interactions in collaborative scenarios by enlighten the opaque relation between security events and payoff. To provide these insights, we introduce a formal unifying mechanism for detecting collaboratively anomalies combined with a mechanism transferring — in an incentive compatible way — recognized deficits for improving prediction and analysis.

3.1 Economic (repeated) Game

An **asset** is anything tangible or intangible that is capable of being owned or controlled to produce value and that is held to have positive economic value. We consider services as the assets. Services might be consumed or provisioned by agents or even by a community or society of multiple agents.

As representation of the involved interacting parties, we introduced the notion of **agents**. Agents have the capability of selecting a **strategy of interactions** based on their beliefs, desires and intensions. Each agent is assumed to make her own

decisions influenced by her intentions, her assumptions, her environment and her constraints to maximize her payoff.

The **admissible strategies** of an agent consists of the consumable services and the set of provided services. The **concrete payoff function** is composed of the value add $v(s)$ of a service invocation, the cost $c(s)$ imposed and the deficit $d(s)$, if the service was executed with security deficiencies

$$p(s) = v(s) - c(s) - d(s).$$

3.2 Agent Categorization

The agents in this game are the service providers and the service consumers. We can categorize agents according to their role within this game as

- **Direct participating agents** contributing to the provisioning or consumption of a service s including the agents that consume or provide services, the service depend or rely on. **Dependent services** are services that are provisioned and consumed for provisioning the service. The set of direct participating agents consists of agents provisioning or consuming these service invocations.
- **Indirect participating agents** having potentially caused or recognized behavior leading to deficiencies. These are potential defenders or attackers. These agents might had already experience in executing deficient strategies.
- **Constructive agents** contribute to the recognition of and counteracts against behavior leading to deficiencies, i.e. defenders.
- **Destructive agents** causing direct or indirect consciously or unconsciously, intended or non-intended behavior leading to deficiencies. These class of course comprises the potential attackers.

Note that this categorization of agents does not assume or imply intent, which is the major difference compared to classical security games. Even a victim of an attack could be classified as destructive. Agents are not a priori divided into attackers or defenders. We intend with this model to identify the experience agents to learn faster and more reliable relations between anomalies, services and deficits. Therefor we will introduce profiles.

3.3 Mechanism Design

Inside the model, **deficits** are assumed to be continuously estimated based on observations made in the past. These continuously and coherently evaluated deficits are transferred to the experienced agents, i.e. agents with a similar profile as a rationale for transfer. This transfer reverses the security risk diffusion, such that deficits will concentrate on the origins of recognition, making the experienced agents witnesses for a collaboratively recognized and shared deficit. It allows to trace back defective service invocations, i.e. assets requiring to be secured.

As a consequence, when transferred deficits exceed, there must be a rationale inside a profile. By investigating an agent's profile one can reason about the deficit and about the severity of the impact on the agent, i.e. the aggregated damages for effected agents and the aggregated damage that were caused by the agent's behavior. And one can estimate the value of the remedies. We expect that mere allocated aggregated deficits will motivate agents to contribute their capabilities and resources to reduce security risks.

This approach resolves the dilemma of the most common assumptions made, namely that risk tolerance will naturally be sanctioned economically. This assumption betrays a misunderstanding of the distributed characteristics of security which is now rectified by the consequent transfers.

Agents have a **profile** of admissible strategies, values and costs, residual value deficits, and transferred deficits. The profile trail corresponds to a reputation. These profiles comprise streams of joint strategy selections of agents that reflect consumption and provisioning of services. Each agent's intentions yield to actions in a special context influencing her payoff.

3.4 Operational Semantics

To enable the estimation of deficits it is necessary to agree on a service invocation semantics, i.e. an agreed model of operations. Consider the service system having some not-observable states and a **transition function**

$$\xi : X \to \Sigma(X),$$

where $\Sigma(X)$ indicating the possible outcomes of taking a transition. Let Σ be the **signature** and X be the **carrier or states**. This behavior model describes the relation of systems and their behaviors in terms of outcomes. An information system together with a starting state x_0 forms a **process**. A starting state is assumed to correspond one-to-one to a service invocation. We use **morphisms** as structure preserving mappings to investigate relationships between systems. This sketched co-algebraic definition is a relaxed form of specification that allows to connect adaptively observation of measurements with transitions, i.e. behaviors.

To make for instance the transitions observed more explicit consider e.g. labeled transition systems. These are a triple $(X, \Lambda, \to)$ out of explicit states X, labels Λ and transitions $\to \subseteq X \times \Lambda \times X$. It can be equivalently formulated as a co-algebra using a power set.

$$\xi_X : X \to 2^{\Lambda \times X} \cong (2^X)^{\Lambda},$$

where $(\lambda, \tau) \in c_X(s) \Leftrightarrow s \to_\lambda t$. The labels $\lambda \in \Lambda$ are considered as observables for transitions τ. We can even identify similar behavior when we extracted explicit states. Let e.g. $\xi : X \to 2^{\Lambda \times X}$ and $\xi' : X' \to 2^{\Lambda \times X'}$ be two co-algebras

of transition systems with the same underlying functor. The homomorphism between these systems is a function $f : X \to X$ such that $2^{\Lambda \times f} \circ \xi = \xi' \circ f$, where $2^{\Lambda \times f} : V \mapsto \{(\lambda, f(s)) \mid (\lambda, s) \in V\}$. This commutation is equivalent to bisimulation. The concrete semantics can embed multiple monitoring systems, i.e. multiple labelings $\mathcal{L} = \{\Lambda_i \mid i \in I\}$, each spawning its own algebra or system $X \to 2^{\Lambda_i \times X}$. These systems embed canonically into the unifying algebra $\xi_X : X \to 2^{\bigcup \mathcal{L} \times X}$.

3.5 Deficit Estimation

Finally, this is a method for formally making behavior explicit allowing to quantify deficiencies following a learning approach. Let (machine) learning be the task of inferring a function from supervised training data. The training data consist of a set of training examples. Each example is a pair (x, y) consisting of behavior x as input values and an associated deficit as output values. A learning algorithm analyzes the training data and produces an inferred function, the regression function $f : x \mapsto y$ with minimal error. We gather training data out of feedback from the experienced agents and derive estimators for deficits. By that we can collaboratively identify and integrate relevant features as a latent semantics that influence deficits by means of regression functions implementing collaborative filtering.

The setting outlined allows to consider the **value of information security** as an economic good. However, the responsibility for ensuring information security is assigned to different domains in the realm of selfish acting agents, i.e. the organizations mentioned in the standards. They naturally follow their economic interests to maximize their payoff when interacting, while interactions in this setting are the provisioning and consumption of services inside the covering social-economic system. We used a relaxed notion of game borrowed from Game Theory, where we assume money as a yardstick for a value maximizing strategy, because it can be easily transferred between agents.

3.6 Profiles

How a specific agent interacts collaboratively (or selfish) should be a rational decision maximizing the specific agent's payoff. The agents will follow their motivational imperative according to their divergent interests. There are two major interest: maximizing the utility for each agent and decreasing deficits. Assume that each agent i intends to follow strategies s maximizing her payoff $p_i(s) = v_i(s) - c_i(s) - d_i(s)$. The communality, i.e. all stakeholders together intends to minimize $\sum_i d_i(s)$ as social welfare. We consider the stream of selected strategies together with impact values as **impact profile**

$$\pi_i = (s_t, v_t, c_t, d_t)_{t \in T}$$

for agent i over time T, where s_t is the selected strategy at time t that implies the values v_t, cost c_t, estimated deficit d_t.

Profiles allow identifying relevant experience as neighborhood. This has the advantages of simplicity, justifiability, efficiency and stability with respect to changing profile arrangements and changes in the adverse environment.

3.7 Relevant Communities and Experience

Consider e.g. the **relevant community of experience** by the similarity of profiles. This community is the set of agents sharing similar deficits when following similar strategies. Let $\langle \pi_i, \pi_j \rangle$ be a corresponding similarity metric, e.g. the Cosine Vector similarity $CV(x,y) = \frac{x^T y}{\|x\| \cdot \|y\|}$ or the Pearson Correlation $PC(x,y) = \frac{(x-E(x))(y-E(y))}{\sqrt{(x-E(x))^2 (y-E(y))^2}}$.

Similarity metrics enables us also to concentrate deficits to experienced agents. These are the agents having the information necessary in their profile for deriving countermeasures. Collaboration needs are indicated by transferring deficits to those agents that can influence value deficits. We thus concentrate diffuse liability onto experienced agents by transferring the deficit d_i of agent i at time t to the agents j having a similar profile, e.g. according to the rule

$$a_j = d_i \frac{\langle \pi_i, \pi_j \rangle}{\sum_j \langle \pi_i, \pi_j \rangle}.$$

requiring to extend the profile information by **transfer profile** $\tau_i = (s_t, a_t)_{t \in T}$ that indicates the experienced agents.

Regression models allow to integrate different data sources in their model equations without a priory knowledge. Continuous adaptation gained from the feedbacks establish a latent semantic. Since expected deficit of agent i becomes predictable, recommendations can be derived based on profile information π_i and the collaboratively established regression estimators. An agent can be advised to use a service with a lower risk profile. This contributes to the decrease of deficits and hence to increased security. Concretely this could be realized by maximizing the agent's utility reflecting cost, risk, and deficit. For any community C of agents and a defined time interval Ξ, there is a canonical profile aggregation

$$\coprod_{i \in C, t \in \Xi} \pi_i(t) \oplus \tau_i(t),$$

where the deficits are summed up per strategy, i.e. attached discrete information. These aggregations constitute a view for the community on relevant model elements and their associated risk. Note that the deficits are exactly the observed security risks. By linearity of the used operators — mainly the expectation value and linear estimators, the model allows to integrate values by summation. Since we have used only linear mappings and plain set theory, we can aggregate over any model

component allowing to investigate evolution of any service, agent, security guard, semantic structure, etc. This allows profound data mining and root cause analysis directed by explicit deficiencies.

4 Discussion and Conclusions

Information infrastructures have become an essential and ubiquitos factor in our economic and social environment. Therefore ensuring information security of these infrastructures is of increasing concern to our society.

The collaborative treatment of information security as an common economic good contributing to social welfare offers multiple dimensions of improvement. An economic approach is fostering collaboration between involved parties even in competitive scenarios and can overcome barriers of misaligned incentives to comply with minimum securtiy baselines.

The importance of collaboration among diffenrent domains, i.e. legislation and service provider, is already addressed on a fundamental level. Serveral activities of the European Commission and of ENISA approaches the exchange of information concering information security incidents and emergencies between established emergency teams of governments and private organizations.

On a long-term perspective, it is vital that organizations participating in such frameworks experience quantitative, economic payoffs of contributing their knowledge and capabilities. Today, there is lack of precise and reliable evidence. Hence economics of information security are still subject to comprehensive research delivering valuable insights.

Acknowledgments

We would like to thank our company for the freedom to conduct this work and our colleagues for listening with patience to the ideas. This document was created partially in the context of the project "Attack analysis and Security concepts for MObile Network infrastructures, supported by collaborative Information exchAnge (ASMONIA)", see http://www.asmonia.de.

References

1. Anderson, R., Böhme, R., Clayton, R., and Moore, T.: Security economics and the internal market. Report to the European Network and Information Security Agency (ENISA), 2007.
2. German Federal Agency for Information Security: IT-Grundschutzkataloge. Bonn (2011)
3. European Commission: Communication from the Commission on Critical Infrastructure Protection - Achievements and next steps: Towards global cyber security. Official Journal of the European Union, COM (2011) 163 Brussels (2011)

4. European Commission: Communication from the Commission on Critical Infrastructure Protection - Protecting Europe form large scale cyber attacks and disruptions: enhancing prepardness, security and resilience. Official Journal of the European Union, COM (200) 149, Brussels (2009)
5. European Commission: Regulatory framework for electronic communications in the European Union. Information Society and Media Directorarte-General, Brussels (2010)
6. European Network and Information Security Agency: Baseline capabilities for national / governmental CERTs - Operational Aspects. Heraklion (2009)
7. European Network and Information Security Agency: Baseline capabilities for national / governmental CERTs - Policy Recommendations. Heraklion (2009)
8. European Network and Information Security Agency: CERT Cooperation and its further facilitation by relevant stakeholders. Heraklion (2006)
9. European Network and Information Security Agency: Cyber Europe 2010 - Evaluation Report. Heraklion (2020)
10. European Network and Information Security Agency: EISAS - European Information Sharing and Alert System for citizens and SME's - Implementation trough cooperation. Heraklion (2011)
11. European Network and Information Security Agency: Incentives and Challenges for Information Sharing in the Context of Network and Information Security. Heraklion (2010)
12. European Network and Information Security Agency: Priorities for Reseach on Current and Emerging Network Technologies. Heraklion (2010)
13. European Network and Information Security Agency: Proactive detection of network security incidents. Heraklion (2011)
14. International Organization for Standardization: ISO/IEC 27000:2011 - Information technology - Security techniques - Overview an vocabulary. Geneva (2011)
15. International Organization for Standardization: ISO/IEC 27001:2008-09 - Information technology - Security techniques - Information security managment systems - Requirements. Geneva (2009)
16. International Organization for Standardization: ISO/IEC 27005:2008 - Information technology - Security techniques - Information security risk management. Geneva (2008)
17. International Organization for Standardization: ISO/IEC 27010:2012 - Information technology - Security techniques - Information security management for inter-sector and inter-organizational communications, Geneva (2012)
18. International Organization for Standardization: ISO/IEC 27035:2011 - Information technology - Security techniques - Information security incident management. Geneva (2011)
19. Moore, T. and Anderson, R.: Economics and Internet Security: a Survey of Recent Analytical, Empirical and Behavioral Research. In: Peitz, M., Waldfogel, J. (Eds.), The Oxford Handbook of the Digital Economy, Oxford University Press 2011.
20. National Aeronautics and Space Administration: NASA Risk-Informed Decision Making Handbook. Version 1.0, Washington D.C. (2010)
21. United States National Institute of Standards and Technology: NIST Special Publication 800-30 - Risk Management Guide for Information Technology Systems, Gaithersburg (MD) (2002)
22. United States National Institute of Standards and Technology: NIST Special Publication 800-61, Revision 1 - Computer Security Incident Handling Guide. Gaithersburg (MD) (2008
23. United States National Institute of Standards and Technology: NIST Special Publication 800-137 - Information Security Continuous Monitoring for Federal Information Systems and Organizations. Gaithersburg (MD)(2011)
24. Schechter, S.E.: Computer Security Strength & Risk: A Quantitative Approach. In: Ph.D. Thesis, Harvard University, Cambridge, 2004.

Index

Securing cyber space first choice are preventive measures to reduce risks. Today, however, systems also protected by detecting anomalies or exceptional situations and predictive conclusions. Such IT Early Warning Systems enable pro-active security measures, improve situational awareness or suggest action in exceptional situations. Addressing researcher and early adopters, recent advances in current research and open issues are presented.

ISBN 978-3-8396-0474-8