Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM) Focal Topic: Advanced Persistent Threats

Summer Semester 2017

Munich, Germany

Editors

Georg Carle, Daniel Raumer, Lukas Schwaighofer

Publisher

Chair of Network Architectures and Services

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Proceedings zu den Seminaren Future Internet (FI) und Innovative Internet-Technologien und Mobilkommunikation (IITM) Themenschwerpunkt: Advanced Persistent Threats

Sommersemester 2017

München, 24. 02. 2017 – 16. 08. 2017

Editoren: Georg Carle, Daniel Raumer, Lukas Schwaighofer

Network Architectures
and Services
NET 2017-09-1

Proceedings of the Seminars
Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)
Summer Semester 2017

Editors:

Georg Carle
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
Technische Universität München
85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: `https://net.in.tum.de/~carle/`

Daniel Raumer
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: raumer@net.in.tum.de
Internet: `https://net.in.tum.de/~raumer/`

Lukas Schwaighofer
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: schwaighofer@net.in.tum.de
Internet: `https://net.in.tum.de/~schwaighofer/`

Cataloging-in-Publication Data

Seminars FI & IITM SS 17
Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM)
München, Germany, 24. 02. 2017 – 16. 08. 2017
ISBN: 978-3-937201-58-0

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2017-09-1
Lehrstuhl für Netzarchitekturen und Netzdienste (I8) NET 2017-09-1
Series Editor: Georg Carle, Technische Universität München, Germany

Vorwort

Vor Ihnen liegen die Proceedings der beiden Seminare „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM), welche die finalen Ausarbeitungen unserer Studierenden enthalten. Die beiden Seminare wurden am Lehrstuhl für Netzarchitekturen und Netzdienste an der Fakultät für Informatik der Technischen Universität München im Sommersemester 2017 durchgeführt. Alle veröffentlichten Ausarbeitungen wurden in englischer Sprache verfasst.

In einer Zeit, in der eine immer größer werdende Vernetzung von persönlicher und kritischer Infrastruktur zu beobachten ist, kommt es gehäuft zu Angriffen, die sich durch einen technisch fortgeschrittenen und auf das angegriffene System spezialisierten Ablauf auszeichnen. Solche, typischerweise als Advanced Persistent Threats bezeichneten Angriffe, werden oft so verdeckt und vorsichtig durchgeführt, dass sie kaum unmittelbar bemerkt werden. Auf diese Weise können Angriffe auf physische Anlagen aus Bereichen wie zum Beispiel Stromversorgung oder chemischer Industrie durch Manipulation von Steuerungsanlagen vorbereitet werden. Ebenso können aus Institutionen über Monate oder sogar Jahre hinweg sensible Geheimnisse entwendet werden. Um dieser Entwicklung gerecht zu werden, gab es in diesem Semester neben vielen anderen aktuellen Themen aus unserem Forschungsbereich erstmals einen eigenen Themenblock, der sich mit dieser neuen Klasse von Angriffen befasst.

Unter den Teilnehmern der Seminare verliehen wir, wie in jedem Semester, einen Best Paper Award. Im IITM-Seminar ging dieser an Herrn Helge Brügner, der in seiner Ausarbeitung „Holt-Winters Traffic Prediction on Aggregated Flow Data“ die Eignung des Holt-Winters-Algorithmus zur Verkehrsvorhersage analysiert. Die beste Arbeit im FI-Seminar wurde im Sonderthemenblock Advanced Persistent Threats von Herrn Claes Adam Wendelin verfasst, der in seiner Ausarbeitung „Malware Detection in Secured Systems“, eine Übersicht über Malware und deren Erkennungsmethoden gibt.

Einige der Vorträge wurden aufgezeichnet und sind auf unserem Medienportal unter `https://media.net.in.tum.de` abrufbar.

Im FI-Seminar wurden Beiträge zu den folgenden Themen verfasst:

- Performanzmodellierung von Netzwerkgeräten
- Privatheiterhaltende Mechanismen für Geoinformationen
- Erkennung von Loadbalancern im Internet

Auf `https://media.net.in.tum.de/#%23Future%20Internet%23SS17` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Im IITM-Seminar wurden die folgenden Themen abgedeckt:

- Holt-Winters Verfahren zur Verkehrsvorhersage mit aggregierten Verkehrsflussdaten
- Finanzielle Anreize in PKI-Infrastrukturen mittels Blockchains
- Eine Einführung in öffentliche und private Distributed Ledgers
- Warteschlangentheorie zur Modellierung softwarebasierter Paketverarbeitungssysteme
- Anomalieerkennung in Netzwerken

Auf `https://media.net.in.tum.de/#%23IITM%23SS17` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Im Sonderthemenblock Advanced Persistent Threats wurden die folgenden Themen behandelt:

- Verteidigungsmechanismen gegen Advanced Persistent Threats
- Malwarestrategien in Cyber-Konflikten
- Erkennung von Schadsoftware in sicheren Systemen

Wir hoffen, dass Sie den Beiträgen dieser Seminare wertvolle Anregungen entnehmen können. Falls Sie weiteres Interesse an unseren Arbeiten haben, so finden Sie weitere Informationen auf unserer Homepage `https://net.in.tum.de`.

München, September 2017

Georg Carle

Daniel Raumer

Lukas Schwaighofer

Preface

We are pleased to present you the proceedings of our seminars on "Future Internet" (FI) and "Innovative Internet Technologies and Mobile Communication" (IITM) which took place in the Summer Semester 2017. Today, an increasing number of devices from personal private networks and critical infrastructure is connected to the global and public network. A new class of attacks on those systems can be observed: specifically tailored to the system and with a high technical level those attacks last for months or even years. Continuous data leaks or malicious changes in the system behaviour cause significant damage until they are discovered. For the first time, we offered a special session that dealt with the so-called Advanced Persistent Threats. All published papers are written in English.

We honoured the best paper of each seminar with an award. This semester the award in the IITM seminar was given to Mr. Helge Brügner who analysed the Holt-Winters algorithm for its applicability to traffic prediction in his paper "Holt-Winters Traffic Prediction on Aggregated Flow Data". In the FI seminar, the award was given to Mr. Claes Wendelin for his paper "Malware Detection in Secured Systems" wherein he presents an overview to malware and the mechanisms to detect it.

Some of the talks were recorded and published on our media portal `https://media.net.in.tum.de`.

In the seminar FI we dealt with issues and innovations in network research. The following topics were covered:

- Performance Models of Network Interconnect Devices
- Location Privacy Preserving Mechanisms
- Detecting load balancers in the Internet

Recordings can be accessed on `https://media.net.in.tum.de/#%23Future%20Internet%23SS17`.

In the seminar IITM we dealt with different topics in the area of network technologies, including mobile communication networks. The following topics were covered:

- Holt-Winters Traffic Prediction on Aggregated Flow Data
- Using the blockchain to add automated financial incentives to the Public Key Infrastructure
- An introduction to Public and Private Distributed Ledgers
- Challenges for Modelling of Software-based Packet Processing in Commodity-Hardware using Queueing Theory
- Network Anomaly Detection

Recordings can be accessed on `https://media.net.in.tum.de/#%23IITM%23SS17`.

In the special session on Advanced Persistent Threats the following topics were covered:

- Defense Techniques against Advanced Persistent Threats
- Strategies for Malware in Cyber Conflicts
- Malware Detection in Secured Systems

We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, September 2017

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany

Seminarleitung

Daniel Raumer, Technische Universität München, Germany
Lukas Schwaighofer, Technische Universität München, Germany

Betreuer

Stefan Liebald (liebald@net.in.tum.de)
Technische Universität München

Holger Kinkelin (kinkelin@net.in.tum.de)
Technische Universität München

Jochen Kögel (jochen.koegel@isarnet.de)
IsarNet Software Solutions GmbH

Marcel von Maltitz (maltitz@net.in.tum.de)
Technische Universität München

Johannes Naab (naab@net.in.tum.de)
Technische Universität München

Heiko Niedermayer (niedermayer@net.in.tum.de)
Technische Universität München

Daniel Raumer (raumer@net.in.tum.de)
Technische Universität München

Minoo Rouhi (rouhi@net.in.tum.de)
Technische Universität München

Dominik Scholz (scholz@net.in.tum.de)
Technische Universität München

Sree Harsha Totakura (totakura@net.in.tum.de)
Technische Universität München

Seminarhomepage

`https://net.in.tum.de/teaching/ss17/seminars/`

Inhaltsverzeichnis

Future Internet

Innovative Internet-Technologien und Mobilkommunikation

Sonderthemenblock Advanced Persistent Threats

A Taxonomy of Performance Models of Network Interconnect Devices

Joline Bui
Supervisor: Daniel Raumer
Seminar Future Internet SS2017
Chair of Network Architectures and Services
Departement of Informatics, Technical University of Munich
Email: Joline.Bui@tum.de

ABSTRACT

A major objective of computer network performance and optimization models is to reduce energy consumption as much as possible, thus lowering costs for electricity and the CO_2 footprint while still offering a high service quality to the user. To this extent, the study of queueing packets within a network, as well as resource modeling, workload modeling, and task scheduling are considered suitable to explain the basics of some network performance models. The presented models are compared to each other in terms of their performance measurement aspects. Furthermore, the comparison of several modeling techniques shows that the selection of a model or a model technique depends on the expected results, parametrized constraints, and the execution in real world systems.

Keywords

Queueing Theory, Network Calculus, Resource Modeling, Workload Modeling, Task Scheduling Problem, List Scheduling, Dynamic Programming, Linear Programming, Integer Programming, Simplex Method, Branch and Bound.

1. INTRODUCTION

The research and study of quality of service guarantees has been motivated by the increasing demand in transmitting multimedia and other real time applications over the Internet [25]. *Packet switching* and *circuit switching* are two networking methods for transferring data between two nodes or hosts. In packet-switched networks the route is not exclusively determined. Routing algorithms are used, which allows many users to share the same data path in the network. This type of communication between sender and receiver is called *connectionless* [39]. On the other hand, in circuit-switched networks the route is setup and established prior to initializing connections to the host. Circuit-switched networks are used in telephone systems [49]. In contrast, most traffic over the Internet uses *packet switching* because the Internet is basically a connectionless network [39] and because it performs better than circuit switching in terms of average delay and buffer requirements [2]. For the communication channels in the network, queues are formed inside the switching nodes, such as routers, bridges, and switches [33]. Many aspects significantly influence the performance within many networks and they can be modeled with various performance models.

In this paper, the focus is on three major fields of computer networks: The first field deals with the study of queueing of packets and its performance models. The second field covers *workload modeling* and the third field is concerned with the *task scheduling* if many processors are available in a network node for parallel programming. Hence, taxonomies of performance models of network interconnect devices are presented. These give instructions under which conditions which models should be selected. First, section 2 gives insights into the *queueing theory* and queueing systems. Then section 3 compares two performance model techniques which deal with queueing type problems encountered in computer networks: *queueing theory* and *network calculus*. Section 4 covers *resource modeling* and subsequently *workload modeling* is presented in section 5. Finally, model techniques for schedule optimization and task allocation within computer networks are presented in section 6.

2. QUEUEING THEORY

2.1 Background Information

The origin of *queueing theory* can be traced back to the early 1900s when A. K. Erlang, a Danish engineer, applied this theory extensively to study the behaviour of telephone networks [41]. In a network node, tasks are queued for CPU in various stages of their processing [41]. Queueing systems are dynamic and contain a flow of network data. There are two different types of flows [29]:

1. ***Steady flow***: These systems contain exact numbers of constraints and do not depend on time. For example, there is a network of channels, each with a channel capacity. To analyze those systems, graph theory, combinatorial mathematics, optimization theory, mathematical programming and heuristic programming are used to deal with scheduling problem and task allocation. This will be explained in more detail in section 6.
2. ***Unsteady flow***: These systems do not have exact numbers of constraints and must be predicted. Stochastic tools are often used in this context and will be described further in section 3.

Queueing theory can be clustered into three major components: input process, system structure and output process [41]. These components are shown in Figure 1. The input process, also called the arrival process, deals with three aspects: the size of the arriving population, the arriving pattern of the customers and the behaviour of the arriving customers. The system structure is concerned with the systems resources, which is described by the physical number and layout of servers, as well as constraints like the system capacity. The output process, also called the departure

doi: 10.2313/NET-2017-09-1_01

process, is characterized by the type of queueing discipline and the service-time distribution. These characteristics are demonstrated in Figure 2 as they contribute to modeled performance aspects.

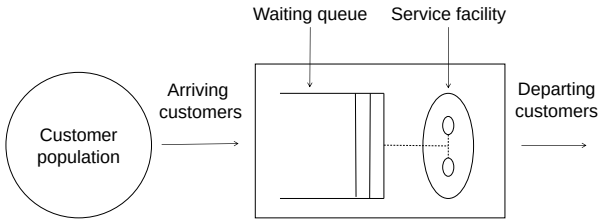


Figure 1: Schematic diagram of a queueing system [41]

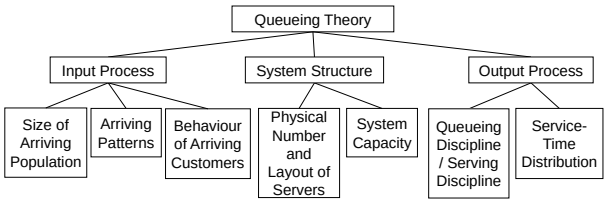


Figure 2: Components and characteristics of queueing theory [41]

The following components, derived from the characteristics presented in Figure 2, can be calculated with various mathematical tools [41]:

- average number of customers in the system
- distribution of number of customers in the system
- average waiting time of a customer in the system
- distribution of waiting time
- length of busy period
- length of an idle period
- current work backlog expressed in units of time

These components represent performance measurement aspects that vary in quantity in different queueing systems.

2.2 Kendall Notation

Queueing systems can have various compilations of different elements. They are described as shorthand notations, known as the *Kendall notation*, which was developed by David G. Kendall, a British statistician, to describe a queueing system containing a single waiting queue which has been further extented [28]. The following notations apply:

A / B / X / Y / Z

A : Customer arriving pattern (inter-arrival-time distribution)
B : Service pattern (service-time distribution)
X : Number of parallel servers
Y : System capacity
Z : Queueing discipline

Example: The classical queue $M/M/1/\infty/FCFS$, but known as $M/M/1$, represents a queuing system where customers arrive according to *Poisson process* and request exponentially distributed service times from the server. The system has only one server, an infinite waiting queue and customers are served on a First Come First Serve (FCFS) basis. If only the first three parameters are shown, the default values for the last two parameters are Y = ∞ and Z = FCFS.

2.3 Queueing Systems

Queueing systems are clustered into two groups of systems, which have many variations of further queueing characteristics within them [41]:

Markovian queueing system: A markovian queueing system is characterized by a *Poisson* arrival process and exponentially distributed service times. Both the arrival and service process are *memoryless*.

Semi-markovian queueing system: Semi-markovian queueing systems refer to those queueing systems in which either the arrival or service process is not *memoryless*. They include $M/G/1$, $G/M/1$, their priority variants and the multi-server counterparts.

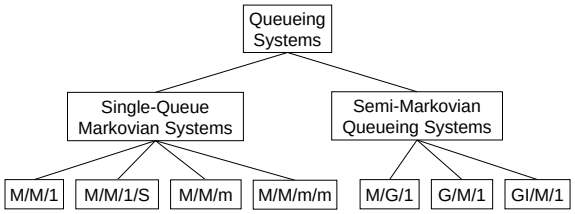


Figure 3: Components and characteristics of queueing theory [41]

Figure 3 gives an overview of the two groups of systems and basic types of queueing systems, whereas $M/M/1$ is referred to as the classical queueing system. For detailed explanation of the other queueing systems that are presented in Figure 3, [41] gives clear definitions and demarcations. Queueing networks are collections of interactive queueing systems, whereby departures of some queues enter some other queues. This can happen deterministically or probabilistically. From the network topology point of view, queueing networks can be categorized into two classes [41]:

Open queueing networks: In this queueing network, customers arrive from external sources outside the domain of interest, go through several queues, or even revisit a certain queue more than once and finally leave the system. Open queueing networks are good models for analyzing circuit-switching and packet-switching data networks.

Closed queueing networks: In this queueing network, customers do not arrive at or depart from the system. There is a constant number of customers that behave within the network like in the open queueing networks. Other than open queueing networks, closed queueing networks are good models for analyzing window-type network flow controls as well as CPU task scheduling problems.

3. QUEUEING THEORY COMPARED TO NETWORK CALCULUS

Generally, *queueing theory* considers the average quantities in an equilibrium state and the traffic that enters a queueing system (or queueing network) is always characterized by a stochastic process, such as the *Poisson* distri-

doi: 10.2313/NET-2017-09-1_01

bution. However, unique customer and service characteristics and requirements in such networks as the Internet often make the application difficult to analyze [26]. Beginning with [15], it has been shown several times that the *Poisson* distribution is not necessarily a realistic assumption for Internet traffic [43]. In contrast, in network calculus the arrival traffic is assumed to be *unknown* as long as it satisfies the following regularity constraints: "The amount of work brought along by the arriving customers within a time interval is less than a value that depends on the length of that interval" [41]. *Network calculus* is a new paradigm of queueing analysis, which is part of the deterministic queueing [35]. This paradigm is able to put deterministic bounds on network performance measures. This new methodology was pioneered by Rene L. Cruz [10] for analyzing delay and buffering requirements of network elements and was further developed by Jean-Yves Boudec and Thiran [35]. The main difference between these two is that *queueing theory* allows us to make statistical predictions of performance measures whereas *network calculus* establishes deterministic bounds on performance guarantees. The idea of *network calculus* is that service guarantees can be achieved by regulating the traffic and deterministic scheduling. Analog to the classical *queueing theory*, a system is classified into the same major components as classical queueing theory (see Figure 1). The input, mostly referred to as an arrival curve, is an abstraction of the traffic regulation, and the transfer function, mostly referred to as a service curve, is an abstraction of the scheduling [43]. The difference with classical *queueing theory* is that non-traditional algebra, such as *min-plus-algebra* and *max-plus-algebra*, is used to transform complex network systems into analytically tractable systems, as well as focusing on the worst case [26]. *Network calculus* has developed into two branches:

Deterministic network calculus: R. L. Cruz introduced in [10] the idea of using a function to deterministically upper-bound the cumulative arrival process. For arrival modeling he developed the arrival curve model, which specifies a traffic envelope to arrivals. In contrast, for server modeling a function to deterministically lower-bound the cumulative service process is used [44],[26]. However, the service curve model evolved and out of it emerged the idea to compare the actual departure time with a virtual time function and to use the difference together with the rate parameter to define the *virtual clock scheduling* algorithm [57],[26]. A server model, called *guaranteed rate* was developed in [18],[26]. Based on these concepts, the derivation of worst-case performance bounds including backlog and delay can be calculated [14].

Stochastic network calculus: Latest techniques try to bring stochastic approaches into *network calculus* [3]. The arrival and server models of stochastic *network calculus* can be considered as probabilistic extension or counterparts of the deterministic *network calculus* [26]. The approach's goal is to comprehend statistical multiplexing and scheduling of non-trivial traffic sources in a framework for end-to-end analysis of multi-node networks [14]. Fidler stated in [14] that compared to classical *queueing theory*, the stochastic *network calculus* comprises a much larger variety of stochastic processes, including long range dependent, self-similar [47], and heavy-tailed traffic [36].

The backlog as a function of time is shown as the vertical marked as b(t) in Figure 4. It is the amount of bits that is held inside the system. The virtual delay at time t is the

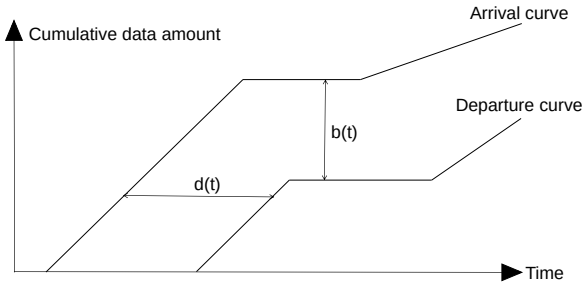


Figure 4: Example of an input and an output function with horizontal and vertical deviations, that demonstrate delay and backlog in network calculus theory [35, Page 5]

delay that would be experienced by a bit arriving at time t if all bits received before it are served before it. It is the horizontal deviation, marked as d(t) in Figure 4 [34].

4. RESOURCE MODELING

In this section, further performance models describe resources of which limitations can be analyzed. Packet processing systems make extensive use of resources like cashes, memory controllers, buses and NICs [12]. In [16] the performance of the packet processing application built for high-speed IO frameworks is modeled. As resources have limits and bottlenecks, in [16] four distinctive characteristics that limit the performance of the packet processing are listed:

1. The maximum transfer rate of the used NICs
2. The link capacity, such as the capacity of a PCI express, which is used to connect the NICs to the rest of the system
3. The restriction of the possible network bandwidth in a RAM
4. The maximum processing load on the CPU, which can be kept low due to modern offloading features of NICs

For these characteristics, the high-performance prediction model can be used to provide upper bounds for the capabilities of a software-based packet processing system [16]. According to Rizzo [48], packet processing costs can be divided into per-byte and per-packet costs, whereas the latter has a deeper impact on IO frameworks. This model assumes that if the limit of the NIC is reached and a constant load per packet times the number of packets is lower than the available CPU cycles, the cycles are spent waiting for new packets in the busy wait loop. By adding these costs, a new value of the constant cost per packet is calculated [16]. The constant cost per packet consists of the sum of the cost used for sending and receiving packets (c_{IO}), the cost of the complexity of the task (c_{task}) and the cost that is introduced by the busy wait on sending or receiving packets (c_{busy}). The available CPU cycles are the main limiting factor of software packet processing. The throughput of a packet processing application heavily depends on the amount of CPU cycles available for its processing task. According to [16], this amount is influenced by the following factors for real world applications:

doi: 10.2313/NET-2017-09-1_01

- the complexity of packet processing
- the time the CPU spends waiting for data to arrive in cache
- the effect of different batch sizes

These factors can be determined precisely by various measurement methods. Experiments in [16] show following performance characteristics predicted by this model: Further measurements demonstrate that this model can be applied to estimate processing tasks, which can be approximated with a constant average load. A possible use case for this model is to evaluate the eligibility of PC systems for specific packet processing tasks. In addition, the results illustrate the trade-off between throughput and latency with different queue sizes. The larger the batch sizes, the bigger the throughput but also the higher the average latency, even though performance increases. The reason is, that packets spend more time queued the larger the batch sizes are. However, batch sizes also have a lower bound due to overloading frameworks. Hence, smaller batch sizes are more suitable for applications that are sensitive to high latency, whereas larger batch sizes are more suitable for application where raw performance is critical.

5. WORKLOAD MODELING

As the resource model describes how to deal with resource limits, workload modeling gives insights about characteristics of the load, which have an impact on resource modeling as well as on task scheduling.
"The workload behavior of the system can be defined as the disposition of resource usage over a period of time. The disposition allows us to characterize the behavior of the system during that period of time [23]."

As shown in Figure 5, workload modeling can be classi-

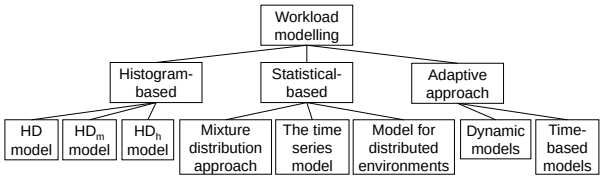


Figure 5: Taxonomy of workload modelling [23],[4],[46]

fied in three different branches: models that are based on histograms, models that are statistical-based, and adaptive approaches. Workloads can be visualized as histograms [23].

The *histogram model* [53], [50], was introduced by Skelly to predict buffer occupancy and loss rate for multiplexed streams. The loss rate and network delay distribution can be obtained using the buffer occupancy distribution [23]. In [23] three traffic models are proposed:

1. *HD* ***model*** [53],[50]: This model has only one histogram for a given sample period and is short-range dependent. It uses an analysis method based on a M/D/1/N queueing system. Compared to the other models and to the real traffic, this model reduces the number of cells. Although, the inaccuracy increases with the number of cells, the model is simple and compact and the results are still accurate. Therefore, it is a good approximation.

2. HD_m ***model*** [38]: This method is based on the modification of the Mean Value Analysis (MVA) algorithm. It is based on several histograms using different time scales. The disadvantage of this model is that they resolve each histogram's groups independently. Therefore, no dependencies between the histograms groups are taken into account. Furthermore, the selection of the number and values of the sample periods can be one problem. The model can be too complex with too many sample periods. It is more accurate but is not as compact as the *HD model*. But compared to MVA, the HD_m loss curve is more precise.

3. HD_h ***model*** [22]: This model is based on a *Hurst parameter* and is long-range dependent. The underlying idea is to modify the variance of the histogram depending on the *Hurst parameter*. In addition, this model needs only one histogram.

The results of the experiment in [23] showed that HD_m has the best results. However, this approach can be cumbersome. Therefore, the best approach is the HD_h *model*, which is compact and very precise.

In general, loss ratio and node delay are traffic quality of service parameters that are obtained. So, the probability that a packet is delayed is higher than a certain value can be predicted. For example, in application areas like video and audio transmission, the delay histogram can be useful in the end nodes to adapt their transmissions rates or to configure the buffer in the reception nodes. Further application areas are e.g., admission control, traffic provisioning, and network configuration [23].

The main objective of workload models is the optimal provisioning of network resources, so service cost of network transmission can be reduced. According to [4], "the workload is considered to be only that processing which is specifically required to satisfy the user request".

In [4], three models are presented, which are used to characterize the system workload and are constructed in a statistical framework: Therefore, any user request R made to a distributed computing system may be characterized by the quadruple (T, L, X, F): T = time, L = location, X = amount of service requested, F = flag.

1. ***A mixture distribution approach*** (A): This model takes only X and F into account and ignores the effects of time and spatial distribution of the requests. The advantage of this formulation is that a very complex probability distribution can be represented in terms of several simpler distributions. When no reasonable assumptions about the conditional probability distributions can be made, clustering is a viable approach for construction of probabilistic models of workload. All clustering techniques assume that the items to be grouped are specified in terms of certain parameters. Therefore, a workload characterization study using the clustering approach can be effected significantly by judicious choice of such parameters [4].

Stochastic process workload models (B, C): The goal is to create a validated probabilistic workload model first, and then to use that model to construct a test workload [4].

2. ***The time series model*** (B): This model takes the time characteristics of the requests into account and ignores the source of the requests. Time series analysis requires data be collected over a long period of time. For example, the number of jobs processed per day may be adequate to analyze the load on a system over a year. To study the time-

doi: 10.2313/NET-2017-09-1_01

dependent nature of steps in the job a markovian model can be used [4].

3. ***Model for distributed environments*** (C): The requests of this model are characterized by the quadruple. If location is allowed to have only a few discrete values, then this model may be considered to be a multivariate point process over time and space. In an actual distributed computing environment, more complex models may be required. The techniques of multivariate point process analysis are likely to be useful. These models may then be used for generating representative test workloads [4].

The third branch of workload modeling in Figure 5 are ***adaptive workload models***:
These are models that can learn and adapt to any environment. There are dynamic models [21] and time-based models [7]. In [46], the adaptive model, Support Vector Regression (SVR) model, predicts the short-term future as a string. The information content in the string conveys as a substring that reflects the future. It has been successfully applied in real environments, such as web traffic generation and virtual path bandwidth management in ATM networks. Therefore, these models could also be applied in areas such as MPEG video stream modeling, intrusion detection, and intelligent prefetching [46].

6. SCHEDULE OPTIMIZATION AND TASK ALLOCATION

In chapter 2, CPU task scheduling was mentioned in terms of *closed queueing networks*. This chapter deals with task scheduling problems in multiprocessor systems. The increasing demand for efficient parallel programming algorithms leads to this issue gaining importance [54]. The *multiprocessor scheduling theory* is concerned with optimally allocating sets of agents to complete a set of tasks over time [54]. Scheduling is a fundamental hard problem (an *NP-hard* optimization problem [31]), "as the time needed to solve it optimally grows exponentially with the number of tasks" [55]. In related works, e.g. [37],[42],[20],[45],[52],[56],[9],[24], good but not optimal scheduling algorithms are presented [55]. Optimal schedules can make a fundamental difference e.g. for time critical systems, or to enable the precise evaluation of scheduling heuristics [55].

When computing an optimal schedule, the system may be trying to achieve one or more of the following objectives [54]:

- minimize the service time of the schedule
- minimize the weighed completion time of each service
- minimize the total number of processors used to solve the problem
- minimize the amount of communication bandwidth used
- if there is a value associated with assigning a task to a particular processor, maximize the aggregate value
- if there is a cost (other than time) associated with assigning a task to a particular processor, minimize the aggregate cost

In general, schedules are constrained by the execution delay of each service, the task release times and precedence relations, as well as communication delays between processors [54]. As many variations of scheduling problems exist, many techniques were proposed to solve them. There are some techniques that assume that processors are a scarce resource, while others suppose the opposite. Some techniques assume that the execution time for each task on each processor is constant [13], while others do not consider the communication delay, which implies that the communication time between processors is zero [8]. Some fundamental scheduling techniques are presented in the Figure 6 to give and overview of the different approaches:

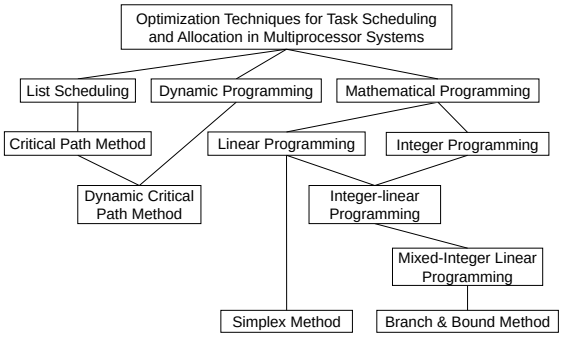


Figure 6: Optimization techniques for task scheduling and allocation in multiprocessor systems [54]

List scheduling works with priority levels by which tasks are ordered in a list. Therefore, the first step is the task prioritizing phase, where a priority is computed and assigned to each task. The second step is the processor selection phase, where the prioritized tasks are assigned to processors that minimizes a suitable cost function. The processor selection phase can be static or dynamic:

Static: The two phases happen sequentially. Hence, the processor selection phase starts after the completion of the task prioritizing phase [5],[54].

Dynamic: The processor selection phase and the task prioritizing phase are interleaved [19],[11].

Most scheduling heuristics algorithms are based on *list scheduling* [5],[6],[8],[54], but they differ mainly in different prioritizations of the tasks. However, the main goal of all *list scheduling* algorithms is to find and place the most critical tasks high on the priority list, so they can be executed quickly [55]. For example, the *critical path* method is a technique, where priorities are assigned to tasks according to how far away they are from the starting point, therefore giving preference to tasks that are likely to be on the *critical path*. This is the longest chain of tasks, where its length is a lower bound on the optimal schedule that can be produced [54]. Ronald Graham, a mathematician also known for his work in *scheduling theory*, first examined *list scheduling*, a stochastic approximation, in detail in 1966 [19]. He looked at a classification that assumes that each processor is identical, therefore execution delay is a function of the task only but no communication delay between the processors is considered [54]. By assuming the latter, Kohler [30] and Sih [51] pointed out, that this can be a problem, because large communication delays between processors can often result in sub-optimal makespan [54]. R. E. Bellmann, the scientist who coined the term *dynamic programming* developed the theory in its initial stages [5]. *Dynamic programming* is a technique that breaks the scheduling problem into simpler sub-problems at different points in time. By getting optimal solutions to the smaller sub-problems, optimal solutions to the bigger scheduling problem can be found [1]. A variation of the introduced *critical path* method is the *dynamic critical path* method [17], that works on following steps [32]:

doi: 10.2313/NET-2017-09-1_01

1. Determine the *critical path* of the task graph and select the next node to be scheduled dynamically.
2. Rearrange the schedule on each processor dynamically. The positions of the nodes in the partial schedules are not fixed until all nodes have been considered.
3. Select a suitable processor for a node by looking ahead the potential starting times of the remaining nodes on that processor.

According to [54] "*dynamic programming* may be used to develop polynomial-time approximation algorithms or optimal solutions for very simple scheduling classifications". However, it is only suitable to solve schedules optimally if it runs in exponential time, which is no better than mixed *integer linear programming* techniques [54]. The third branch in the Figure 6 is *mathematical programming*. This tool solves complex problems such as those that can be modeled as an objective function with a set of mathematical constraints [54]. Generally, the objective is to minimize or maximize a linear function of these variables. *Linear programming* (LP) problems are *mathematical programming* formulations where the objective and each constraint is formulated as a linear function of X_1, X_2,... , X_n. Therefore, an LP-formulation would have the following structure of constraints [54]:

$MIN(orMAX)$:
$c_1X_1 + c_2X_2 + ... + c_nX_n$

$Subject\ to$:
$a_{11}X_1 + a_{12}X_2 + ... + a_{1n}X_n \leq \mathrm{b}_1$
...
$a_{k_1}X_1 + a_{k_2}X_2 + ... + a_{k_n}X_n \geq \mathrm{b}_k$
...
$a_{m_1}X_1 + a_{m_2}X_2 + ... + a_{m_n}X_n = b_m$

One widely used algorithm for solving *linear programming* problems is the *simplex method*. It has been proven to find optimal solutions (if there are any) to all instances of *linear programming* problems. The underlying idea of the algorithm is that there is always a single optimal solution to any *linear programming* problem, which can be found on a corner point of the feasible region [6]. Another method of *mathematical programming* is *integer programming*, in which some or all the variables are restricted to being integer. *Integer linear programming* (ILP) adds the rule, that the objective function and the constraints must be linear. These models are for cases, where the decision variable represents a nonfractional entity such as people or cars, or where a decision variable is needed to model a logical statement. For instance, 5.23 cars cannot be produced - only 5 or 6. Another example is to set integers for binary decisions, such as yes (1) and no (0).

For multiprocessor scheduling with communication delays (MSPCD) optimal ILP solutions are presented in [55] which aim to maximize the utilisation of each available processor and minimize the task schedule length. To achieve this goal, the algorithms need to find where and when each task will be executed, so that the total completion time is minimal. Node costs as required time for task completion, edge costs as communication time between two tasks on different processors as well as data transfer times as communication delays are considered as constraints. A variation of LP-formulations based on diverse logic systems can be derived from this provided information. In [55] the formulation of ILP-RevisedBooleanLogic (ILP-RBL) and ILP-TranssivityClause (ILP-TC) are proposed. While the formulation ILP-RBL reworked the logic for Boolean multiplication, ILP-TC enforces the partial ordering of the processor indices with the help of an additional transivity clause. Both formulation methods manage to reduce the number of variables and constraints by effective linearization of the bilinear equation arising out of communication delays in the MSPCD model. The results of the experimental evaluation show that in the ILP-RBL the formulation runs faster over a small numbers of processors and the linearization in ILP-TC runs faster over a larger number of processors.

Other cases exist where some decision variables must be integers but the remaining decision variables are allowed to be non-integers. In those cases, *mixed integer-linear programming* can be applied [54]. The *branch and bound* technique is used to find those integer solutions [54]. The idea behind *branch and bound* (B&B) is to optimally solve the problem via a relaxation of the problem. A relaxation is a simplification of a problem, which makes the calculation easier.

In [27] a parametrized B&B algorithm for scheduling real-time tasks on a multiprocessor system is presented to minimize the maximum task lateness in the system. Task lateness is defined as the difference between a task's completion time and its deadline. The algorithm uses a search tree that represents the solution space of the problem. Each vertex in the search tree represents one specific task-to-processor assignment with schedule ordering. The optimal solution is presented as one or many vertices (if one exist), where all tasks have been scheduled on the processors. With the aid of intelligent rules for selecting vertices to explore/expand and pruning vertices that do not lead to an optimal solution, the complexity of the search can be reduced. However, not only the choice of a vertex selection rule, e.g. last-in-first-out (LIFO) or least-lower-bound (LLB), but also the determination of a lower-bound funtion and an approximation strategy of the B&B method, have impacts on the optimal solution.

In a maximization problem, the optimal objective function value of an ILP-relaxation is always an upper bound on the optimal integer solution, whereas any integer feasible point found is always a lower bound on the optimal integer solution [6],[54]. The values are used to compare and update the upper and lower bounds at every step to narrow down the solution area, so the optimal integer solution is found [54]. Tompkins stated in [54] that *branch and bound* works better for *mixed-integer-linear programming* problems than listing every possible integer solution. One problem of *branch and bound* is that the running time in some cases may grow exponentially because of the size of the problem. However, he stated that "the closer the linear programming relaxation is to the bound, the less time it takes to find the optimal integer solution, as less division can be made" [54].

7. CONCLUSION AND OUTLOOK

Many different parts within a computer network can be modeled to give information about the network performance. Throughout this paper, many complex models and techniques are presented to obtain specific results to improve network performance. The challenge here is to figure out

what results are expected and how constraints need to be parametrized. Furthermore, models and techniques often do not provide very precise results. Therefore, taxonomies are presented to propose subordinate topics which lead to techniques that match the problem and present their advantages and disadvantages.

In the future, optimization problems within queueing and scheduling in networks will become more relevant due to the increasing amount of data and therefore also increasing data traffic. The long-term objective is to save as much energy as possible, thus reducing the electricity costs and lowering the CO_2 footprint. This leads towards the concept of *Green NFV Infrastructure*, which deals with placing as many virtual network functions as possible on the smallest set of physical servers [40]. Here as well, optimization models, such as mixed-integer optimization, with e.g. the *robust optimization technique*, are used to solve the virtual network function placement problem to optimality [40]. Therefore, more complex performance models that are used in many areas of different layers of a network are required and gain importance.

8. REFERENCES

[1] Dynamic Programming. Available online at `https://www.cs.cmu.edu/~avrim/451f09/lectures/lect1001.pdf`; last accessed on 2017/04/02.

[2] Queueing delay. Available online at `http://www.cs.toronto.edu/~marbach/COURSES/CSC358_S14/delay.pdf`; last accessed on 2017/04/01.

[3] Systemperformanz. Available online at `https://www.net.in.tum.de/pub/systemperformanz/ss2012/skript/networkcalculus.pdf`; last accessed on 2017/04/01.

[4] A. K. Agrawala, J. M. Mohr, and R. M. Bryant. An approach to the workload characterization problem. *Computer*, 9(6):18–32, 1976.

[5] R. Bellman. Dynamic programming (dp). 1957.

[6] S. Bradley, A. Hax, and T. Magnanti. Applied mathematical programming. 1977.

[7] M. Calzarossa and G. Serazzi. A characterization of the variation in time of workload arrival patterns. *IEEE Transactions on Computers*, C-34(2):156–162, Feb 1985.

[8] C. Chekuri and M. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41(2):212–224, 2001.

[9] E. G. Coffman and R. L. Graham. Optimal scheduling for two-processor systems. *Acta informatica*, 1(3):200–213, 1972.

[10] R. L. Cruz. A calculus for network delay. i. network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, 1991.

[11] O. Deutsch, M. B. Adams, and J. Lepanto. Closed-loop hierarchical control of military air operations. Technical report, DTIC Document, 2002.

[12] M. Dobrescu, K. Argyraki, and S. Ratnasamy. Toward predictable performance in software packet-processing platforms. Technical report, 2012.

[13] D. W. Engels, J. Feldman, D. R. Karger, and M. Ruhl. Parallel processor scheduling with delay constraints. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 577–585. Society for Industrial and Applied Mathematics, 2001.

[14] M. Fidler and A. Rizk. A guide to the stochastic network calculus. *IEEE Communications Surveys & Tutorials*, 17(1):92–105, 2015.

[15] J. L. Florin Ciucu, Almut Burchard. A network service curve approach for the stochastic analysis of networks. *SIGMETRICS Perform. Eval. Rev.*, 2005.

[16] S. Gallenmueller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle. Comparison of frameworks for high-performance packet io. In *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*, pages 29–38. IEEE, 2015.

[17] M. J. Gonzalez Jr. Deterministic processor scheduling. *ACM Computing Surveys (CSUR)*, 9(3):173–204, 1977.

[18] P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 273–284. Springer, 1995.

[19] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Labs Technical Journal*, 45(9):1563–1581, 1966.

[20] T. Hagras and J. Janeček. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. *Parallel Computing*, 31(7):653–670, 2005.

[21] G. Haring. *On state-dependent workload characterization by software resources*, volume 11. ACM, 1982.

[22] O. Hashida, Y. Takahashi, and S. Shimogawa. Switched bath bernoulli process (sbbp) and the discrete-time sbbp/g/1 queue with application to statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, 9(3):394–401, 1991.

[23] E. Hernández-Orallo and J. Vila-Carbó. Network performance analysis based on histogram workload models. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th International Symposium on*, pages 209–216. IEEE, 2007.

[24] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee. Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, 18(2):244–257, 1989.

[25] Y. Jiang. A basic stochastic network calculus. In *SIGCOMM Comput. Commun. Rev.*, 2006.

[26] Y. Jiang. Network calculus and queueing theory: Two sides of one coin: Invited paper. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 37:1–37:12, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[27] J. Jonsson and K. G. Shin. A parametrized branch-and-bound strategy for scheduling precedence-constrained tasks on a multiprocessor system. In *Parallel Processing, 1997., Proceedings of the 1997 International Conference on*, pages 158–165. IEEE, 1997.

[28] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[29] L. Kleinrock. Queueing systems, volume i: theory. 1975.

[30] W. H. Kohler. A preliminary evaluation of the critical path method for scheduling tasks on multiprocessor systems. *IEEE Transactions on Computers*, 100(12):1235–1238, 1975.

[31] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE transactions on parallel and distributed systems*, 7(5):506–521, 1996.

[32] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE transactions on parallel and distributed systems*, 7(5):506–521, 1996.

[33] S. S. Lam and J. Wong. *Queueing network models of packet switching networks*. Faculty of Mathematics, University of Waterloo, 1981.

[34] J.-Y. Le Boudec and P. Thiran. A short tutorial on network calculus. i. fundamental bounds in communication networks. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 93–96. IEEE, 2000.

[35] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

[36] J. Liebeherr, A. Burchard, and F. Ciucu. Delay bounds in communication networks with heavy-tailed and self-similar traffic. *IEEE Transactions on Information Theory*, 58(2):1010–1024, 2012.

[37] W. Löwe and W. Zimmermann. Scheduling iterative programs onto logp-machine. *Euro-ParâĂŹ99 Parallel Processing*, pages 332–339, 1999.

[38] J. Luthi, S. Majumdar, and G. Haring. Mean value analysis for computer systems with variabilities in workload. In *Computer Performance and Dependability Symposium, 1996., Proceedings of IEEE International*, pages 32–41. IEEE, 1996.

[39] Margaret Rouse. packet-switched. Available online at `http://searchnetworking.techtarget.com/definition/packet-switched`; last accessed on 2017/04/01.

[40] A. Marotta and A. Kassler. A power efficient and robust virtual network functions placement problem. In *Teletraffic Congress (ITC 28), 2016 28th International*, volume 1, pages 331–339. IEEE, 2016.

[41] C.-H. Ng and S. Boon-Hee. *Queueing modelling fundamentals: With applications in communication networks*. John Wiley & Sons, 2008.

[42] A. Palmer and O. Sinnen. Scheduling algorithm based on force directed clustering. In *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*, pages 311–318. IEEE, 2008.

[43] K. Pandit, J. Schmittt, and R. Steinmetz. Network calculus meets queueing theory-a simulation based approach to bounded queues. In *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on*, pages 114–120. IEEE, 2004.

[44] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM transactions on networking*, 1(3):344–357, 1993.

[45] A. Radulescu and A. J. Van Gemund. Low-cost task scheduling for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 13(6):648–658, 2002.

[46] S. Raghavan, N. Swaminathan, and J. Srinivasan. Predicting behavior patterns using adaptive workload models [computer networks]. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999. Proceedings. 7th International Symposium on*, pages 226–233. IEEE, 1999.

[47] A. Rizk and M. Fidler. Non-asymptotic end-to-end performance bounds for networks with long range dependent fbm cross traffic. *Computer Networks*, 56(1):127–141, 2012.

[48] L. Rizzo. Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.

[49] M. Rouse. circuit-switched. Available online at `http://searchnetworking.techtarget.com/definition/circuit-switched`; last accessed on 2017/04/01.

[50] N. Schroff and M. Schwartz. Video modeling within networks using deterministic smoothing at the source. In *INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE*, pages 342–349. IEEE, 1994.

[51] G. C. Sih and E. A. Lee. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE transactions on Parallel and Distributed systems*, 4(2):175–187, 1993.

[52] O. Sinnen. *Task scheduling for parallel systems*, volume 60. John Wiley & Sons, 2007.

[53] P. Skelly, M. Schwartz, and S. Dixit. A histogram-based model for video traffic behavior in an atm multiplexer. *IEEE/ACM Transactions on Networking (TON)*, 1(4):446–459, 1993.

[54] M. F. Tompkins. *Optimization techniques for task allocation and scheduling in distributed multi-agent operations*. PhD thesis, Massachusetts Institute of Technology, 2003.

[55] S. Venugopalan and O. Sinnen. Optimal linear programming solutions for multiprocessor scheduling with communication delays. *Algorithms and architectures for parallel processing*, pages 129–138, 2012.

[56] T. Yang and A. Gerasoulis. List scheduling with and without communication delays. *Parallel Computing*, 19(12):1321–1344, 1993.

[57] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *ACM SIGCOMM Computer Communication Review*, volume 20, pages 19–29. ACM, 1990.

Location Privacy Preserving Mechanisms

Friederike Groschupp
Betreuer: Sree Harsha Totakura
Seminar Future Internet SS2017
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: friederike.groschupp@tum.de

ABSTRACT
Location privacy is not an issue every user thinks of when using Google Maps, a fitness tracker or his GPS navigation system. Yet location privacy is an important part of one's anonymity while using location-based services. If location privacy is compromised, an adversary can draw sensitive conclusions about the user or use the gained information for his advantage. In the recent years, different approaches have been developed aiming to provide location privacy for different use cases. This paper aims to give an overview over the functionality, features and drawbacks of several important approaches: cloaking, mix zones, dummy queries, peer-to-peer systems and private information retrieval.

Keywords
Location-Based Services, Location Privacy, Anonymity

1. INTRODUCTION
A Location-Based Service (LBS) is an application that uses geographical information provided by a user in order to offer a service [12]. Today, LBSs are widely spread and used e.g. for vehicle or pedestrian navigation, providing location-based information or receiving a service at a specified area.

LBSs have become more popular and are used more often, resulting in users disclosing their location information more often. It is important to think about the user's information security and the threat to his anonymity. With the location information disclosed, an adversary, that is suspected at the LBS, might be able to draw conclusions about a user's lifestyle. Solely the fact that it is known that one was at a specific location can be bothersome. Additionally, anonymous messages sent from a location can be matched to the sender if it is known that they were present at the time. If the user has revealed their position knowingly or unknowingly in a previous message and now sends a message that is supposed to be anonymous and contains the same location information, the two messages and therefore the identity can be linked.

This paper discusses several approaches to preserve location privacy. At first, the problem is described in section 2, together with the use cases of LBSs, the protection goals, the assumptions about the system model, the adversary and the threats. Section 3 briefly introduces k-anonymity and location servers, two basic concepts used for location privacy. The location privacy approaches itself are discussed in Section 4: Temporal and spatial cloaking (4.1), mix zones (4.2) and dummy queries (4.3), together with the security they provide and possible attacks against them. A brief overview over peer-to-peer systems and private information retrieval is given in sections 4.4 and 4.5, respectively.

2. PROBLEM STATEMENT
The goal of this paper is to present an overview of the state-of-the-art approaches to provide location privacy. Location privacy is defined as the capability of precluding other parties than the ones trusted from learning the user's current or former location [1].

The main concern while discussing the issue of location privacy is on the part of the LBS. It is assumed that a LBS logs the received service requests containing location information. As a LBS is seen as a non-trusted party, the goal of location privacy-preserving mechanisms is to prevent accumulation of identifiable location information.

2.1 Use-case Scenarios
In order to discuss approaches aiming to protect location privacy, it is important to understand in what scenarios users may request a service from a LBS. The use cases for LBSs are diverse; some of them are mentioned below. Note that this list is not exhaustive.

- **Retrieving location specific information:** Users often want to gather information about their surroundings, be it finding a good restaurant nearby or the closest hospital or retrieving the weather forecast. Many of those requests are nearest-neighbor or range queries [2]. The technology most used for obtaining the location information for this kind of queries is the Global Positioning System (GPS).
- **Route planning and traffic information:** The use of a GPS-based guidance system while traveling along routes has become a frequently used application. While route planning and guiding is used for journeys the user is unfamiliar with, services like current traffic updates or hazard warnings are also used for often frequented routes [11].
- **Place bound use of services:** Recently, applications whose service is triggered when the user enters a certain area are becoming more popular. An example of such applications is a reward system, which offers discounts when a user enters a certain shop [1]. While

doi: 10.2313/NET-2017-09-1_02

some of these applications are based on GPS information, pervasive computing tools can also be used.

As we see, the situations in which users reveal their location may have very distinct characteristics and user goals. Every situation brings different problems with it. Therefore, the mechanism used in order to preserve location privacy has to fit the requirements of each situation.

2.2 Protection Goals

It is assumed that with a single message containing location information L it is not possible to draw conclusions about the user. This means that the sender of the message cannot be identified due to L. Other information like a username or metadata carried by the messages sent from the user to the LBS may, depending on the application, identify the user. Yet this is not the concern of location privacy.

Golle et. al [5] show that user identity can be drawn from several disclosed locations. If an application is able to link several queries containing location information and some of the locations correspond to the user's home and/or workplace, they might be easy to identify. In those cases where the identity of a user is - willingly or unwillingly - revealed, it should not be possible to link subsequent location updates to the user.

2.3 Assumptions

The approaches are based on similar assumptions regarding the system model, the user and the adversary. These assumptions are presented below.

System Model

In the model we discuss, there is no means of locating the user other than by the provided location information. While other methods such as locating the user e.g. by their IP address are available, they either have been deemed inaccurate [6] or are made difficult through proxies or sufficient use of obfuscation. Furthermore, it is always assumed that the user's mobile device comes with a position sensor (i.e. GPS receiver) and the user provides their information willingly and knowingly.

The applications considered in this paper do not need the user's real identity for providing their service; they are able to work with pseudonyms or without any user-related information.

Adversary

The adversary we regard is the LBS, be it either that the service provider is hostile or its system has been compromised. The user's device is deemed trustworthy, meaning that no malicious software or attacker has access to the position information on the device.

The following assumptions about the adversary apply to the mechanisms presented [11]:

- **Adversary cannot access client's identifiers:** It is assumed that the adversary has no access to the client's network addresses, e.g. their IP address, or needs other client specific information, like a username. This information is sufficiently protected, e.g. through the use of an anonymizing network or through the use of proxies. He is only able to observe the location information provided in the service requests.

- **Adversary can be active or passive:** When an adversary is passive, it only observes the location information provided in the queries and tries to draw conclusions from it.

  In contrast, an active adversary may modify the content of the responses from the LBS in order to trigger a certain behavior of the client. With this change of behavior, it might be simpler to deduce which of the locations is the real one. An additional strategy of an active adversary is to spoof false user information into the system in order to tamper with the results of the anonymization process.

- **Adversary has statistical background knowledge:** The adversary may have access to statistical knowledge, e.g. traffic densities at different locations. This information can be used to infer the real location.

- **Prior knowledge:** If the adversary monitors a specific user, they might have prior knowledge about the user [6].

2.4 Threats

The main purpose of location privacy is to provide sender anonymity, meaning that the adversary is not able to determine the identity of the originator of a message. When using a LBS, the main problem is the location information that is provided by the user. The requester of a service may be identified by correlating the location information with prior knowledge or easily researched information, like someone's home or workplace address.

When a user S sends a Message M containing location information L to a LBS that is the adversary A, sender anonymity and location information are threatened in the following ways [6]:

- **Restricted space identification:** If A knows that only S can be at L (e.g. L is in the area of a residential home), A can conclude that S is in L and has sent M. A trivial search in telephone books or property listings can reveal S's real identity.

- **Observation identification:** If A knows S is positioned at L and observes a message sent from the area of L, A can infer that S may have sent the message. If the user has revealed their identity and location in a former message, a subsequent anonymous message can be linked to it via the location.

- **Location tracking:** If A knows that S was or is at location L_i and has a linked series of location updates $L_1, L_2, \ldots, L_i, \ldots, L_n$, A knows that S visited all of these n locations. S might not want some of the locations they visited to be linked to them.

doi: 10.2313/NET-2017-09-1_02

3. BASIC CONCEPTS

Some of the approaches are based on the same underlying concepts, which are k-anonymity and a trusted third party. These two concepts are briefly introduced below.

k-anonymity

k-anonymity states that within a specific set, a user is indistinguishable from at least $k-1$ other users [9]. In other words, a set is k-anonymous if it includes the user and at least $k-1$ other users identical to it in regards of the attributes considered.

A user's location can be represented by a tuple containing several dimensions: $[x_1, x_2]$, $[y_1, y_2]$ and $[t_1, t_2]$ [6]. $[x_1, x_2]$ and $[y_1, y_2]$ describe the two dimensional area in which the user is positioned. This information is always necessary. In addition, the time period in which the user was located in the area can be determined by $[t_1, t_2]$. If $x_1 \neq x_2$, $y_1 \neq y_2$ and $t_1 \neq t_2$, the tuple does not give away the exact information of the user, but only a certain range. The tuple is k-anonymous when the area it describes encompasses the position of the user and at least $k-1$ other user.

When an approach relies on k-anonymity or anonymity sets it is important to note, that anonymity is only provided in regard to location information. Other service-specific information or prior knowledge of the adversary could still identify the user [6]. For example, when Alice and Bob form one anonymity set and the adversary knows their genders, the adversary can infer that a service request for a women's clinic most probably originates from Alice.

Location Server

Some of the privacy approaches assume the existence of a trusted third party (TTP). This TTP is often called a location server (LS). The users of an anonymizing protocol in an area subscribe to the LS. The task of an LS is to receive the location provided by the user and anonymize it according to the selected approach. It then sends the query with the processed information to the LBS on behalf of the user and receives the response [10]. As the response is tailored to provide the service for all possible points in the enlarged, anonymized area, the LS filters the response. It then forwards the accurate response to the user.

The use of a LS in order to achieve location privacy has been viewed critically. The LS is an other third party that is used, and the most difficult step would be to identify a trustworthy instance. Building an infrastructure of trusted LS would take a lot of effort. Additionally, the TTP is a single point of attack for an adversary. If compromised, the attacker gets hold of all data, uncensored [11].

An other issue is that algorithms based on a TTP rely on other users using the same LS. They need to be present in the closer area in order to anonymize properly and still be able to provide sufficient information. Even if the required number of users is present, the location information the LBS receives will never be accurate but always an enlarged area. This may entail an information overhead, as the server has to provide replies for all possible locations encompassed in the area.

But the use of a TTP also brings an advantage for the user with it: The computations needed for the anonymization process are not done by the user. The user is relieved from heavy computations and other algorithm related details.

4. LOCATION PRIVACY APPROACHES

4.1 Cloaking

The concept of spatial and temporal cloaking [6] uses the concept of k-anonymity and a TTP in form of a LS, as introduced in section 3. The idea of the cloaking approach is to construct a tuple with its comprised ranges (space and/or time) as small as possible that is still k-anonymous. With cloaking, the LBS will never receive the exact information but only a certain interval in which the true location information is included. Therefore, it is important to consider the requirements in order to meet a certain performance and quality of results. Gruteser and Grunwald [6] distinguish between different application areas based on their need on exact spatial or temporal information. Increasing the accuracy of one information attribute can be done by decreasing the accuracy of the other.

Spatial Cloaking

The idea of spatial cloaking is computing a so-called cloaked area that encompasses the user and at least $k-1$ other users. This cloaked area is then forwarded to the LBS. One option to implement spatial cloaking is with a quadtree-based algorithm [6]. The general assumption is that by decreasing the level of required accuracy, k-anonymity can be provided in every situation. The algorithm is provided with the user's information, the parameter k_{min}, which determines the minimum size of the anonymity set, the area covered by the anonymizer and the information of all other users in the area. The detailed algorithm is presented in Listing 1. In short, the algorithm quarters the area considered as long as there are at least $k-1$ other users in the same area as the user. The smallest quadrant that still fulfills this constraint is then returned.

Temporal Cloaking

If a more exact spatial information for a service is required, one can make use of temporal cloaking, where the temporal accuracy is reduced. As all users being present in the area in a certain time interval and not just at one point of time are considered, the number of users available for anonymization increases. For temporal cloaking the user request is delayed until k_{min} other users have resided in the area determined by a resolution parameter. The resolution parameter determines how inaccurate the location information is allowed to be. The time range $[t_1, t_2]$ is then computed as following: t_2 is the current time, t_1 the time of the user request minus a random cloaking factor. This random cloaking factor is important, as the original exact information could be derived from t_1 otherwise. The tuple containing spatial and temporal information is then returned.

Accuracy of Results

Due to the lack of appropriate real life data, simulated automotive traffic flow for different city areas based on released detailed transportation data was used to test the algorithm [6]. According to the results of this experiment, the

accuracy achieved by the algorithm differs based the structural characteristics of the area the user is located in. Areas containing highways have a higher density of vehicles, the median resolution in these areas ranges from 30 to 65 meters with $k_{min} = 5$. For areas mainly comprised of less frequented collector streets, the median resolution decreases to 125 to 250 meters for $k_{min} = 5$. For all scenarios, the mean size of the anonymity set computed is approximately 10 users.

If the application requires a certain minimum resolution, temporal cloaking can be added. If, for example, the provided location has to be exact within 15 meters, a time interval of 30 seconds is required in average for highway areas. At least 70 seconds are normally required for collector street areas [6]. This leads to the conclusion that, as expected, resolution is negatively correlated to the anonymity constraint. Additionally, when the service is not critical in regards of time, the spatial resolution can be increased by decreasing the temporal resolution.

That the algorithm is based on quartering the area causes the mean anonymity to be approximately twice the anonymity constraint. This indicates that an improved algorithm with better discretization could yield a better resolution with a lower mean anonymity closer to the anonymity constraint. This could be achieved for example by dividing the area according to each situation or merging areas that do not fulfill the anonymity constraint on their own.

Security Analysis

One potential active attack to circumvent cloaking approaches is by reporting a large number of additional locations to the LS. This can for example be done by the adversary spoofing false requests to the LS. This results in the LS releasing very accurate location information, as the anonymity constraint is fulfilled for a smaller area. However, the LS can be protected by only accepting one location information from each authenticated user. Acquiring a large number of authentication keys/ authenticated users should be made disproportionately expensive for an adversary.

Problems may emerge when several users issue a request at the same time [6]. A critical situation is depicted in figure 1, where circles represent users positioned in areas described by coordinates $([x_1, x_2], [y_1, y_2])$; $x_1, x_2, y_1, y_2 \in 0, 1, 2$. Let us assume that vehicles 1 to 4 use the same LS at the same time for anonymization. Then, the following requests with overlapping location information is received by the LBS:

vehicle 1: $([0, 1], [0, 1])$

vehicle 2: $([1, 2], [0, 1])$

vehicle 3: $([0, 1], [1, 2])$

vehicle 4: $([0, 2], [0, 2])$

The first three vehicles name their position with three adjacent quadrants. The fourth one however issues the request with a quadrant larger than the others that covers them. Now, if all position information was processed with $k_{min} = 3$ by the LS, the adversary can conclude that vehicle 4 issued

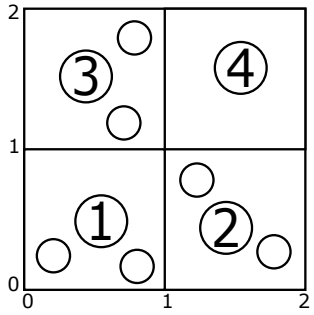


Figure 1: Compromised anonymity through several issued requests. Source: [6]

the request from the quadrant $([1, 2], [1, 2])$. This is because the algorithm would have returned a smaller quadrant otherwise. The k-anonymity constraint can be violated by overlapping simultaneous requests.

4.2 Mix zones

Mix zones are a useful approach when considering pervasive computing scenarios. For this scenario, the user is able to register with location-based applications for callbacks when a user enters a specific zone. The registration is done via a middleware that serves as proxy and as LS. A use case is to register with a service in order to receive advertisements and discounts when entering a shop. Most of these applications are not in need of the user's real identity and therefore able to work with short-term pseudonyms.

The idea of the approach presented by Beresford et. al [1], is to define areas called mix zones. For a group of users the mix zone is defined as a connected spatial region in which a user's position is not known. This is achieved by no user sending location updates. When a user enters a mix zone, all pseudonyms of users within that zone are changed. Within a mix zone, the user identities are "mixed", it cannot be distinguished between different users. Consequently, an observing attacker can not link users coming out of the mix zone with the ones going in.

The areas where users are registered for callbacks are called application zone.

Anonymity Sets

In order to quantify the anonymity provided, the concept of anonymity sets is used [1]. An anonymity set for a user u that visits a mix zone at time period t is the group of people that visit the mix zone during the same period t. The size of this anonymity set offers a first criteria for the anonymity provided. The more people are in it, the more anonymity is provided. The user may want to define a lower threshold, when fewer people are in the anonymity set they might refuse to provide their location in adjacent application zones. Additionally, when the average size of the anonymity sets of the mix zones surrounding an application zone is known, the expected anonymity level can be presented to a user before they sign up for a service provided in the application zone.

```
1. Initialize the quadrants q and q_prev as the total area covered by the anonymizer
2. Initialize a traffic vector v with the current positions of all known vehicles
3. Initialize p as the position of the requestor vehicle
4. If number of vehicles in traffic vector v is less than k_min, return quadrant q_prev
5. Set q_prev to q
6. Divide q into quadrants of equal size
7. Set q to the quadrant that includes p
8. Remove all vehicles outside q from the traffic vector v
9. Repeat from Step 4
```

Listing 1: The spatial cloaking algorithm as presented in [6]. It computes an area containing the requesting user and at least $k-1$ other users.

Security Analysis

Until now, the model and its assumptions about the provided anonymity have presumed that the location of exit is independent to the point of exit. In reality, this is normally not the case [1]. Consider a mix zone with two entry points. When two users enter the mix zone at the two different entry points, they most likely will continue to walk in the same direction they entered the mix zone and exit the zone at the respective opposite site. Only in a very small number of cases, both will turn around and leave the zone through the same point that they entered it. Therefore, the user can be tracked with very high probability.

Entropy

A quantitative metric to measure the degree of anonymity is entropy. The entropy for a mix zone z can be computed based on recorded user movements. For each user traveling through z at time t, the preceding zone p, visited at $t-1$ and the subsequent zone s, visited at $t+1$, are noted. How often each possible pair (p, s) occurred, can be summarized in a movement Matrix M [1].

The probability with which the pair (p, s) occurs, can be calculated by dividing the number of (p, s) in the matrix through the sum of all pairs:

$$P(prec = p, subs = s) = \frac{M(p,s)}{\sum_{i,j} M(i,j)}$$

Ensuing, the conditional probability that the user exits through s, having entered through zone p, can be calculated as:

$$P(subs = s | prec = p) = \frac{M(p,s)}{\sum_{j} M(p,j)}$$

The information content associated with a number of possible outcomes with probability p_i can be calculated with Shannon's classic measure of entropy as:

$$h = -\sum_{i} p_i \cdot \log p_i$$

The lower the entropy is, the more certain an attacker can be about a true answer, and therefore the lower the degree of anonymity will be.

Beresford et. al [1] conducted experiments in their laboratory, where the movement of each staff was tracked with the help of position sensors. They also discuss the same situation as described before, where it is observed that two users enter the mix zones at opposite entrances, and two users leave the mix zones at the respective exit points. According to the data they gathered, the probability that both went straight in the observed situation is 99.9%. The possibility, that they did a U-turn, is 0.1%. The exact data for this results can be found in [1]. The entropy for the above observations can be calculated as

$$h = -(0.999 \cdot \log 0.999 + 0.001 \cdot \log 0.001) = 0.011$$

The entropy is much smaller than 1, therefore, an adversary is able to link pseudonyms and users identities with great success.

If movement possibilities are not equiprobable, an attacker with statistical background knowledge will be able to link user identities with high certainty and the concept of mix zones deems to provide only a low level of anonymity. Only if movement profiles are equally likely or a hostile observer has no statistical background knowledge, mix zones are able to provide anonymity. However, this still requires that the anonymity set is of sufficient size.

4.3 Dummy Queries

One solution to achieve location anonymity without the need of a TTP is the creation of dummy queries. The user does not only request the provided service for one, but for many locations [7]. For a basic dummy-based approach, the service request from the user to the server has the following format:

$$S = \langle u, L_1, L_2, \ldots, L_k \rangle$$

u is a user identifier (that does not permit possibilities to infer the real user identity), $\langle L_1, L_2, \ldots, L_k \rangle$ is a set of $k-1$ dummy locations and the real location. The LBS processes the request s and answers with the response R:

$$R = \langle (L_1, V_1), (L_2, V_2), \ldots, (L_k, V_k) \rangle$$

where V_i is the respective value for L_i requested from the service for $1 \leq i \leq k$. The user then only selects V_r, the value for their real location $L_r, 1 \leq r \leq k$. Note that only the user should know the value of r [7].

These queries have to be constructed carefully, as it would be easy to figure out the false queries when the adversary monitors the user over a longer period of time, uses statistical knowledge or checks the locations for validity (a location in the middle of a lake seems unlikely to be the real position in most cases). Therefore, many thoughts have been given on how to generate realistic dummies. One of the approaches is SybilQuery [11].

SybilQuery is an algorithm presented and proposed by Shankar

et. al [11]. It is fully decentralized and autonomous, based on k-anonymity and aims to provide the creation of realistic dummy queries. In contrast to the approaches described before, it does consider the user traveling along predefined routes.

In the model presented a LBS is a database storing a set of tuples $< l, v >$, where l is a location and v is a value associated with l, like the current traffic condition or points of interest. The user queries the LBS periodically for the values associated with their current location while they move from a starting location to a destination. In order not to reveal their real current location, the users sends $k-1$ additional requests to the LBS. Those queries are generated in accordance with the $k-1$ paths that are created at the beginning of the trip and resemble the real path.

Dummy Query Generation

When the user uses SybilQuery for location privacy preserving, they enter their destination and the security parameter k. SybilQuery then generates $k-1$ synthetic start and end points, with them $k-1$ paths and consequently generates the queries. The details of the process are described below [11].

At first, the endpoints are generated (note that this term is used both for source and destination). In order to generate paths that can not be detected as synthetic paths, a database containing regional traffic statistics is adduced. It is important to note that this is not real-time traffic information, but a statistic of former traffic trends. The endpoint generator uses this data in order to produce endpoints that share characteristics with the original source and destination. Characteristics considered are for example the surrounding traffic density or the likeliness of a location being an endpoint for a trip (a shopping mall would be much more likely than a highway intersection).

As a next step, the path generator uses the k start and end points, which includes the real source and destination, and produces k paths. A path is represented as a sequence of way points. For doing so, it uses a database of regional maps. The path generator and the endpoint generator are only needed once at the beginning of the trip.

After all k paths are constructed, the query generator is triggered with the user's current location. Then, it mimics the user's movement along the real path. Simply spoken, when a user has traveled a certain distance d and sends an other request to the LBS, the generator applies similar offsets to the synthetic paths and sends $k-1$ queries with synthetic locations. However, these simulations are more realistic if for example current traffic conditions are taken into consideration (e.g. if there are traffic congestions along a synthetic path, a slower advance along this path should be simulated).

Extensions

Considering an adversary that may be passive or active and may have statistical background knowledge, the following improvements to the basic principles can make the system more robust towards attacks [11]:

- **Randomized path selection:** A user may not always use the shortest route to their destination. If all synthetic paths correspond to the shortest path between their respective endpoints, the real path is to be figured out easily. Therefore, multiple paths should be generated for each pair of endpoints, the one used for the dummy queries is then chosen with a probabilistic method.
- **Robustness towards active adversary:** An active adversary may report false information like a traffic congestion. A real user may take a detour based on that information. SybilQuery has to mimic this behavior for the synthetic routes.

  Additionally, one way to detect active adversaries is to query multiple LBSs. The responses of the adversary may differ and it can be detected by them.
- **Caching of endpoints:** If the adversary monitors the user over a longer period of time, paths that the user travels frequently (e.g. from their home to their workplace) are easy to identify as the real path. This is because the other parts are generated randomly and therefore do not appear as often. Likewise, when a user finishes a path and starts a new one shortly after, the second set of synthetic paths do no match the first set, when former destination and new source lay too far apart. Those attacks are handled by SybilQuery caching used paths and endpoints.
- **Path continuity:** Even though the generated paths are alike, it is possible for some paths to take longer than the others. If the user has reached their destination before the synthetic paths do and the system stops sending queries, the LBS can spot the synthetic paths. Therefore, SybilQuery continues to imitate movement along the synthetic paths until their destination is reached.
- **GPS sensor noise:** The location provided by the GPS system is normally imprecise. If the locations of the dummy queries are always exactly positioned on the road, the synthetic paths can be detected. Therefore, a random noise is added to the location information for each dummy query.

Security Analysis

The system turns out to be relatively robust to attackers guessing the real path. Shankar et. al [11] conducted a user study, where they presented k paths in the San Francisco Bay Area to volunteers. All volunteers had good knowledge of the area and had location information tools at their disposal. The volunteers had to figure out the real path for $k = 4$ and $k = 6$. The volunteers were able to figure out the real path with a probability of 0.26 for $k = 4$ and 0.19 for $k = 6$. These values are close to the expected values for random guessing (0.25 and 0.17 for $k = 4$ and $k = 6$ respectively). This leads to the conclusion that the synthetic paths generated by SybilQuery resemble real paths.

4.4 Peer-to-Peer Systems

In order to avoid the necessity of a LS, peer-to-peer systems for anonymous location-based queries have been developed. One of those approaches is MobiHide [4], which is based

on k-anonymity. In this approach, the users authenticate with a certification server (which is only used for authentication, not for the anonymization process) and self-organize thereafter. As there is no central LS storing the location of each user, the information has to be stored distributed. The system is based on the Chord [13] distributed hash table architecture. As this method works on one dimension, the user positions are mapped from the two-dimensional space into one-dimensional space, for example with the Hilbert curve. The Hilbert space filling curve is a continuous fractal that can be used for this mapping. The area considered is filled with the Hilbert curve, the location of each user is then determined as the offset of their location along the curve. Each user holds the information about their location in the one dimensional space as well as several pointers towards specified other users. With this architecture, each user is able to determine n neighbored users around him.

When a user wants to request a service from a LBS, MobiHide determines k users including the requester who are consecutive neighbors in the constructed one-dimensional space. Which of the k possible sets is chosen is decided randomly. The user then computes the smallest rectangle that covers all k selected users. The service request containing the area of the rectangle is the sent by the user to the LBS which responds with the respective information. The user then filters the response based on his location for the information he is really looking for.

Security Analysis

MobiHide provides k-anonymity in the case that all users issue queries with the same probability [4]. If this is not the case, the approach is vulnerable to the correlation attack. Consider the extreme case when only one user sends service request. Consequently, the adversary receives multiple rectangles of which it knows that it covers the user's location. The adversary can now intersect the rectangles in order to shrink the number of possibilities for the user's location. However, it is believed that the difference of number of requests per user is not as extreme, even when distribution is not expected to be perfectly even.

4.5 Private Information Retrieval

Private Information Retrieval (PIR) is used when a database is to be queried and the entity holding the database is not to know which entry was queried. One way to achieve this goal would be to present the whole content of the database to the requester and they read the entry they need. Yet this solution is not feasible for reality, as an immense overhead of data traffic is produced and the owner of the database may not want to give away their entire data.

Consequently, computational PIR [8] based on cryptographic assumptions have been developed. Ghinitha et. al [3] propose an algorithm based on the Quadratic Residuosity Assumption. The protocol imposes an overhead of $O(n)$ at the server and costs of $O(\sqrt{n})$ for client-server communication.

A problem that arises for the deployment of PIR schemes is that code modifications at server and client side are required. Therefore, PIR cannot be easily used for existing applications.

Security analysis

In contrast to the other approaches introduced, PIR offers strong cryptographic guarantees on anonymity. It also has significant advantages in regards to the information that is revealed to the LBS [3], as no kind of location information is disclosed. Other approaches submit an inaccurate location or several possible locations. PIR does not reveal any information at all. This is also a protection against correlation attacks.

No information at all is disclosed, this leads to a reduction of the identification probability. Let U be the number of all possible users, e.g. all mobile users in the country. The identification probability for PIR is $\frac{1}{U}$. For the approaches mentioned above, the identification probability is $\frac{1}{k}$, where k denotes the size of the anonymity set or the number of users of the service in the area. Normally, U should be significantly larger than k, implying that the identification probability is significantly lower.

5. CONCLUSION

Several approaches that aim to provide location privacy while using a Location-Based Service have been presented in this paper. The approaches differ in terms of the use cases they are appropriate for, the basic concepts they rely on, the overhead they produce, the accuracy they are able to provide and the attacks they are vulnerable to.

Mix zones are a good choice when working with predefined application areas, for example in pervasive computing. Cloaking methods are appropriate when a decrease of temporal resolution can be tolerated if it increases spatial resolution. Peer-to-peer systems have the advantage of being decentralized and abolishing the need of a TTP. A scheme like SybilQuery that produces dummy queries is convenient to use while traveling along paths and provides independence from other users. Private Information Retrieval is based on cryptographic guarantees and provides the strongest anonymity while producing the most overhead.

Of the location privacy-preserving mechanism presented, none is suited to provide location privacy in every possible use case. For each use case the requirements of the user and the characteristics of the application have to be considered.

For the future it is essential to raise the user's attention towards the importance of location privacy and the dangers when revealing ones location to third parties. In order to be successful, solutions have to be convenient to deploy and easy to use.

6. REFERENCES

[1] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[2] S. Dhar and U. Varshney. Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5):121–128, 2011.

[3] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: Anonymizers are not necessary. *Proceedings*

doi: 10.2313/NET-2017-09-1_02

of the ACM SIGMOD International Conference on Management of Data, 2009.

[4] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Mobihide: A mobilea peer-to-peer system for anonymous location-based queries. In D. Papadias, D. Zhang, and G. Kollios, editors, *Advances in spatial and temporal databases*, volume 4605 of *Lecture Notes in Computer Science*, pages 221–238. Springer, Berlin, 2007.

[5] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In H. Tokuda, M. Beigl, A. Friday, A. J. B. Brush, and Y. Tobe, editors, *Pervasive computing*, volume 5538 of *Lecture Notes in Computer Science*, pages 390–397. Springer, Berlin, 2009.

[6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In D. Siewiorek, M. Baker, and R. T. Morris, editors, *Proceedings of the 1st international conference on Mobile systems, applications and services - MobiSys '03*, pages 31–42, New York, New York, USA, 2003. ACM Press.

[7] C. S. Jensen, H. Lu, and M. L. Yiu. Location privacy techniques in client-server architectures. In C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, editors, *Privacy in Location-Based Applications: Research Issues and Emerging Trends*, pages 31–58. Springer, Berlin, 2009.

[8] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Los Alamitos, Calif., 1997. IEEE Computer Soc. Press.

[9] X. Lu and M. H. Au. Chapter 11 - an introduction to various privacy models. In M. H. Au and K.-K. R. Choo, editors, *Mobile Security and Privacy*. Syngress, Boston, 2017.

[10] G. Myles, A. Friday, and N. Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, 2003.

[11] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In A. A. Helal, editor, *UbiComp '09*, page 31, New York, N.Y, 2009. Association for Computing Machinery.

[12] S. Spiekermann. General aspects of location based services. In J. H. Schiller and A. Voisard, editors, *Location-based services*, Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, San Francisco, CA, 2004.

[13] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

Detecting load balancers in the Internet

Felix Hartmond
Advisor: Minoo Rouhi, Dominik Scholz
Seminar Future Internet SS2017
Chair of Network Architectures and Services
Department of Informatics, Technical University of Munich
Email: felix.hartmond@tum.de

ABSTRACT

Most tools for measurements of internet properties assume a model of the internet where only one path exists between a source and a destination which is taken by all packets send to this destination. With load balancers, which distribute traffic to multiple paths this model changes. This paper takes a look at the changes which have to been done to the classical traceroute to extend it to a tool which can discover multipath topologies. Additional we will look at results of mass-scans of routes and their findings about the overall usage of load balancers in the internet.

Keywords

traceroute, measurement, load balancing, topology, multipath

1. INTRODUCTION

In the traditional model of the internet a packet which is sent to a certain destination address always takes the same path through the internet. Every router on the path of the packet has a forwarding table which has a fixed mapping of destination addresses to output interfaces of the routers over which the address is reachable. To deliver the packet to the destination, the packet is sent always to the output interface which is defined in this table. This model does not apply to the internet in its current state. Nowadays there are special routers, called load balancers [14, 17], which induce the requirement of a changed model.

In contrast to normal routers, load balancers have multiple output interfaces over which the same destination address is reachable. It distributes all packets addressed to this destination over these interfaces which results in multiple routes leading to the same destination. Multiple possible links to the destination have the advantage of increased Network speed. Moreover, bandwidth and route bottlenecks are dissolved. The network resilience is improved since the destination is still reachable if a route fails.

Figure 1 shows a network with a load balancer. Routes in such networks split at a load balancer into parallel paths. The parallel paths merge back together before they reach the destination host at a convergence point.

There are many widely-used tools for running analysis on networks. One very common tool for analyzing which routers are on the path to a destination is *traceroute*. Like many other tools, it is designed on the old model of the internet without parallel routes. When using traceroute in a network with load balancers, different problems occur which make the results of the scan inaccurate. Therefore, a new tool is needed which can work in load balanced networks.

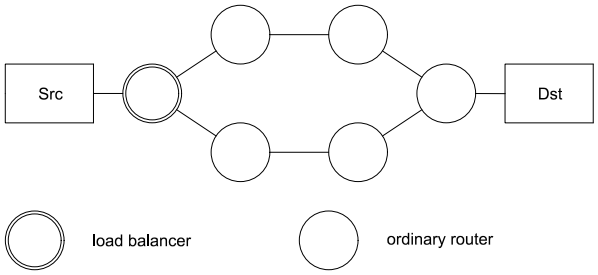


Figure 1: A network with a load balancer

This new tool should be able to identify load balancers on a route and discover the parallel paths behind them. This way the tool can deliver information about the whole topology on the way to the destination address.

Despite analyzing a single path such a tool can also be utilized to investigate the overall usage of load balancers in the internet. This can be done by scanning a lot of different path from different sources to different destinations. If the routes are well distributed over the internet conclusions can be drawn about the overall usage of load balancing in the internet.

There is already a public available implementation of such a multipath detection algorithm called *paris traceroute*. [18]

This paper proceeds as follows. Section 2 defines different types of load balancers. Afterwards in Section 3 we look at the classical traceroute and the necessary steps to extend it to a multipath aware tool like paris traceroute. In the end in Section 4 we look at bulk-scans with such tools which have been done by different researchers and look at their findings about the overall usage of load balancers in the contemporary internet.

2. TYPES OF LOAD BALANCERS

The output interface of a load balancer a packet is sent to is selected based on a so-called *flow identifier*. The flow identifier is calculated from different header fields. Augustin et. al. [6] defined three classes of load balancers which take

doi: 10.2313/NET-2017-09-1_03

different header fields into account:

per-destination. Per-destination load balancers use the destination address of a packet to calculate the flow identifier. This is similar to classic routing however, this balancer assigns different interfaces for different IP addresses in a prefix. All packets addressed to the same host are sent to the same interface. [15]

per-flow. Per-flow load balancers send all packets belonging to the same connection over the same route. Therefore, the IP-5-Tuple [1] is used for the calculation of the flow identifier. Packets from different connections belonging to the same source-destination-pair can be sent over different routes whereas packets belonging to the same connection will always use the same route.

per-packet. Per-packet load balancers do not use any header fields for the calculation of the flow identifier. Every packet is processed independently of the connection it belongs to. The selected output interface can be random or for example based on a round robin principle. Since this processing may lead to packet-reordering inside a connection it can have a bad effect on TCP performance [7]. [15]

When analyzing the usage of load balancing in IPv6 networks Almeida et. al. [2] discovered two additional IPv6-specific types of load balancers:

per-flow with flow label. This type of load balancer works like the already described per-flow balancer. However, instead of using the IP-5-Tupel for the calculation of the flow identifier, it uses the newly introduced flow label field from the IPv6 header. The flow label was introduced as it is less efficient to use the port numbers in IPv6, due to varying offsets of the layer-4-header caused by the new IPv6 extension headers. [9]

per-application. Per-application load balancers are similar to per-destination load balancers. Instead of using the destination addresses they use the destination port to calculate the flow identifier. Thereby the traffic is split according to the application it belongs to.

3. MULTIPATH DETECTION ALGORITHM

To scan networks with load balancers a tool is needed. This section describes the necessary steps to construct an algorithm to analyze a load balanced route. First Section 3.1 looks at the well-known tool traceroute which is widely used to measure paths in networks. The principle of traceroute will be the basis for the tool. Section 3.2 points out different problems of the classic traceroute has when scanning in networks with load balancers. Section 3.3 describes changes which have to be done to the classic traceroute so that consecutive probes for analyzing one possible route are not sent to different routes by a load balancer. Finally in Section 3.4 the changed traceroute is used to reveal all possible paths which packets can take to the scanned destination.

[1]source address, destination address, protocol, source port and destination port

3.1 Classical traceroute

The classical traceroute is a tool to track the route of a packet on its way through a network. To find the intermediate routers between the source and the destination host it sends probe packets addressed to the destination with a low TTL (time to live) header value. It starts with a TTL of one. Every router which forwards a packet decreases the TTL field by one. So the probe packet will have a TTL value of 0 at the first router. This router will drop the probe packet because of that send an ICMP time exceeded back to the source host [11]. Traceroute iteratively increased the TTL of the probe packets until it receives an ICMP port unreachable, which is the answer from the destination host. Traceroute sends per default three probes per hop to get information about each router even when answer packets are lost. Traceroute has different modes which use different types of probes. The traditional mode uses udp packets as probes. There are other methods which use for example ICMP echo requests or TCP SYN packets. [1]

3.2 Problems of the classic traceroute

It is necessary to be able to match the incoming responses to the send probes. The IP header and the first 64 bits of the probe packet are included in an ICMP time exeeded message [11]. The UDP header is only 64 bits long and is completely included in an ICMP time exeeded message. When running with default settings, Traceroute uses the destination port to match the answers and therefore increments the destination port for each sent probe [1].

Like mentioned in Section 2, some types of load balancers include the destination port in the calculation of the flow identifier, that the probe packets of one scan can be distributed to different paths. This leads to different problems and measurement anomalies.

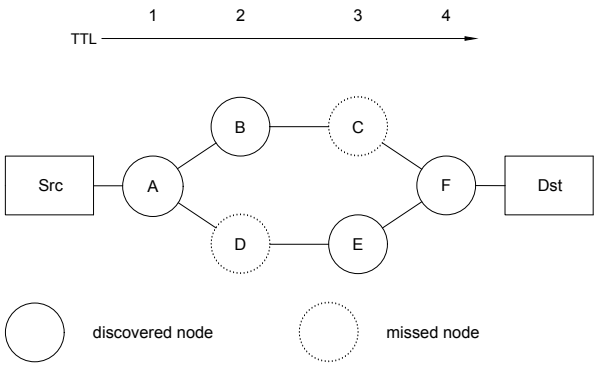


Figure 2: Missing nodes with classical traceroute

One problem is that some of the nodes on the route stay undiscovered. When there are different routers with the same distance from the source only one of them can be reached by a single probe and therefore all other routers at the same distance stay undiscovered. [4]

In the example in Figure 2 the probe with TTL 2 takes the upper route. Router B is detected, but router D, which is also two hops away from the source, stays undiscovered. The same situation appears with the probe with TTL 3 which is

doi: 10.2313/NET-2017-09-1_03

send to the lower route in the example. Router E is found but router C stays undiscovered.

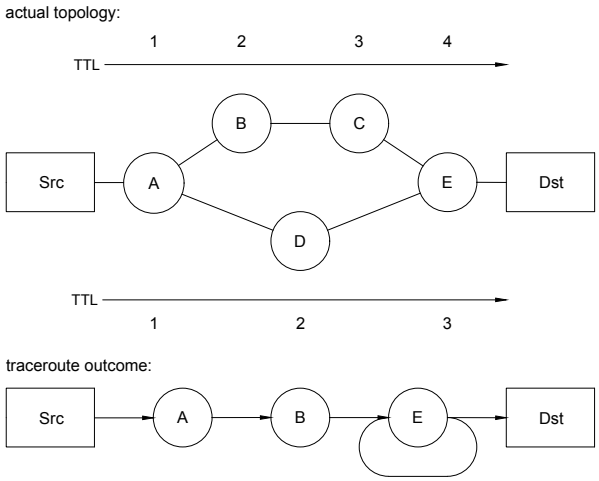


Figure 3: Multiple detection of a router with classical traceroute

Another problem is that router, can be detected multiple times. The result of the traceroute then shows a loop. Such loops can appear if a load balancer distributes probes to routes of different length. With such routes, routers behind the convergence point of the balanced routes are reachable over different distances over the different routes. When a probe takes the short route and the next probe takes a route which is one step longer, the probes end up at the same route which will show up multiple times in the result. In the example in Figure 3 the probe with TTL 3 takes the lower route and the probe with TTL 4 the upper one. They both end up at router E which shows up two times in the result. [4]

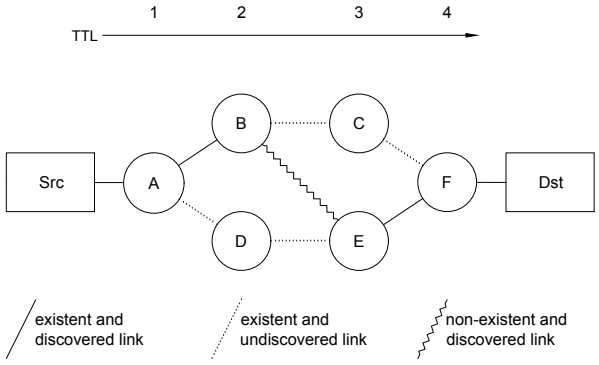


Figure 4: Detection of non-existent links with classical traceroute

A third problem is the detection of non-existing links. The classical traceroute simply draws links between the incrementally discovered routers to show the discovered route. When detected routers are distributed on parallel routes this approach does not work. Routers with a distance difference of one do not have a direct link between them when they are on different routes. In the example in Figure 4 traceroute discovers the routers A, B, E and F. When simply connecting the discovered routers a link is drawn between the routers B and E which does not exist in the topology. [4]

3.3 Controlling the balancers decision

To avoid the problems discussed in the last section traceroute has to be improved to a tool which is aware of the existence of parallel load balanced paths in a scanned network and does not interfere with the existent load balancers without intending this. To find all routers on one of the paths we want all the probes to be sent to the same route by a load balancer, but some fields have to be changed constantly to be able to match the responses to the sent probes.

For each mode of traceroute (UPD, ICMP Echo and TCP) a way is needed to put an identifier into the first 64 bit of the transport header without touching fields which are used to calculate the flow indentifier for any of the load balancer types which were introduces in Section 2.

When scanning the route to a fixed destination, per-destination load balancers do not have to considered because they only use the destination address which is never modified.

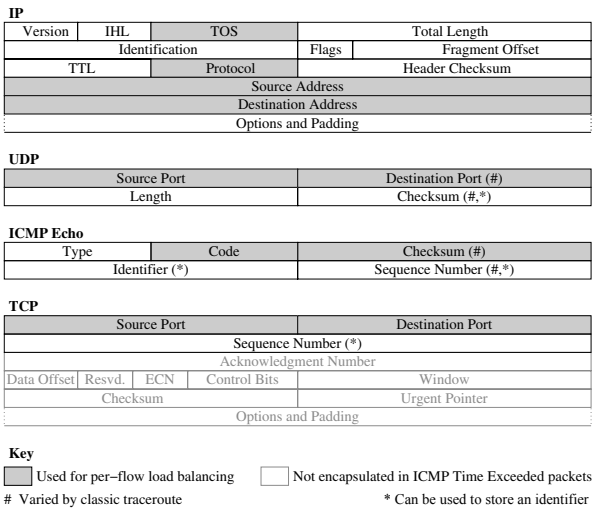


Figure 5: Overview over header fields [4]

Figure 5 summarizes which header fields of the different transport headers are used for per-flow load balancing, which are varied by the classical traceroute and which can be used for storing an identifier without interfering with per-flow load balancers.

For TCP probes the sequence number and for ICMP Echo request probes the indentifier and the sequence number can be easily be set to store an identifier. For UPD probes there are no fields which can take user-defined content, but the checksum field can be used. To vary this field the payload has to be varied to cause different checksums.

Per-packet load balancers can not be controlled due to their nature of handling packets independently from any header fields. Therefore, the tool cannot completely reconstruct a topology which contains per-packet load balancers.

doi: 10.2313/NET-2017-09-1_03

Per-flow balancers which use the IPv6 flow label do not cause problems. When looking at per-flow balancers for IPv4 there already header fields found which can also be used for IPv6 as well. So the flow label field can be left untouched without further problems. The flow label should be initialized with a non-zero value because any router can initialize a flow label which is zero but has to leave a non-zero flow label untouched [3].

To also cover per-applicatoin load balancers no additional changes have to be done. Per-application load blancers use only the destination port field to calculate the flow identifier which was already excluded from the varying fields when covering per-flow balancers.

3.4 Revealing the whole topology

With the changes mentioned in Section 3.3, the modified traceroute is now able to track a route without having probes distributed to different paths by non-per-packet load-balancers. But when tracking a route to a destination we do not want to know only one possible path. Instead, we want to know all possible paths which traffic to the destination can take. To achieve this, the tool has to recognize if there are load balancers on the route and have to track the multiple routes behind them separately. This is only possible for non-per-packet load balancers. Per-packet load balancers can only be detected but due to their non-controllable behavior it is impossible to send probes to a certain path behind them.

Keeping header fields which influence the load balancer's decision steady is very straightforward. But inciting the balancer to send a probe to another output as the previous probe is not. It is not possible to force the probe to be sent to a chosen output interface due to missing information about how the header fields are used and how many outputs the load balancer distributes traffic to.

Additionally, it is unknown which routers are load balancers at all. To identify a router as a load balancer and find all its output interfaces a stochastic approach can be used when sending probes with varying flow identifier.

Because we do not know the exact algorithm of the balancer we can not say if we have found every path for sure. So we have to set a level of confidence which we want to reach with the probing. For probing we assume that all load balancers distribute traffic equally to all output interfaces.

The classical traceroute sends three probes per hop on default [1]. To test if the router is a load balancer this is not enough. With three probes the probability is 25% that all three probes take the same path if the probed router is a load balancer with two output interfaces.

To test if a router is a load balancer it is assumed that the router is a balancer has two output interfaces. Then the tool tries to disprove this assumption with the set level of confidence. For a level of confidence of 95% this needs six probes. If all the six probes take the same way, the tool assumes that the router is a normal router and continues with the next hop. If the probes take more than one route the assumption is set to a load balancer with three output interfaces and more probes are sent to this router. This is continued until an assumption is disproved at the set confidentiality. Scans on the internet have shown that load balancers have up to 16 output interfaces. In such a case, 96 probes are needed to find all interfaces at the confidentiality of 95%. [5]

After identifying a router as a load balancer, it can be tested if the router is a per-flow or a per-packet one. This is possible by assuming a per-packet balancer and sending 6 probes with fixed flow indentifier for a 95% confidentiality of the decision. If all packets go to the same interface the balancer is labeled as a per-flow balancer, otherwise it is labeled as a per-packet balancer. [5]

When a per-packet load balancer is detected it is not possible to reveal the whole topology. But at least all output interfaces of the known routers are revealed.

The tool iteratively probes all new discovered routers and draws the topology of these routers and their paths. When probing a router on a path behind a load balancer first flow identifiers have to be found which reach the tested router. Therefore, a number of identified and those are selected who reach the router which should be probed [6].

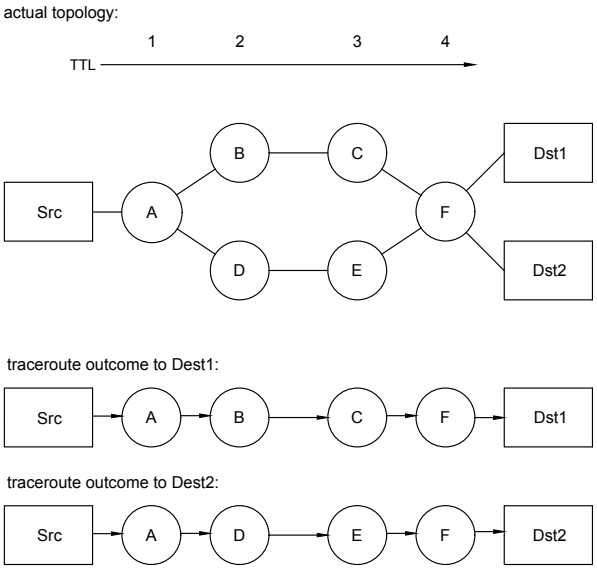


Figure 6: Per-destination load balancing [5]

So far, the tool does not detect per-destination load balancers. To extend the tool to also cover per-packet load balancers it has to trace towards a subnet instead of a single destination address. In the example in Figure 6 node A is a per-destination load balancer. If tracing toward address *Dst1* only the upper path is discovered and only the lower path if tracing towards destination *Dst2*.

To also detect per-destination load balancers the tool creates also flow identifiers which differ in the destination inside the destination prefix address when creating probes for testing if a router is a load balancer.

doi: 10.2313/NET-2017-09-1_03

4. USAGE OF LOAD BALANCERS

To analyze the overall usage of load balancing in the internet different scans [2, 5] have been done with multipath detection algorithms.

Metrics which can be used to describe topologies which are caused by load balancers will be covered in Section 4.1. Section 4.2 takes a look at a method which can be used to realize scans to gain information about the overall usage of load balancing in the internet. Finally, Section 4.3 presents the results the researchers got when executing and evaluating their scans.

4.1 Metrics

Augustin et. al. defined diamonds to describe the structure of load balanced paths. A diamond is defined as a subgraph of the path which begins at a divergence point and ends, two or more hops later, at a convergence point. All possible flows from the source to the destination have to traverse both divergence and convergence point. [6]

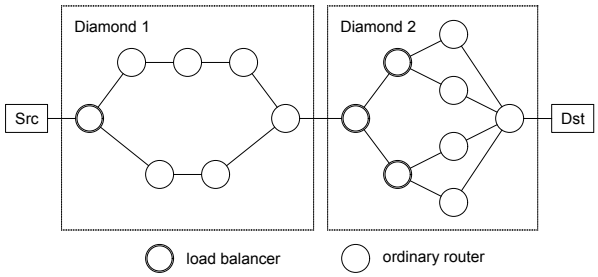


Figure 7: A Network with two diamonds and four load balancers [6]

Figure 7 shows an example of a route with two diamonds. A route can contain multiple diamonds and a diamond can contain multiple load balancers. In the example diamond 2 contains three load balancers.

Further Augustin et. al. [6] defined three metrics to quantize such diamonds:

Diamond width. To describe the diamond width two metrics are used. The min-width describes the amount of link-disjoint paths between the divergence and convergence points. This defines a lower limit for the path diversity inside the diamond. In the example, both diamonds have a min-width of 2. Additionally the max-width describes the maximum number of interfaces which can be reached at a given distance form the source. Diamond 1 has a max-width of 2 and diamond 2 has a max-width of 4.

Diamond length. The diamond length describes the maximum number of hops between divergence and convergence point. In the example Diamond 1 has the length 4 and diamond 2 has the length 3.

Diamond symmetry. A diamond is considered symmetric if all possible path are of the same length. If not, the diamond is asymmetric and the asymmetry describes the difference between the longest and the shortest path. In the example, diamond 1 is asymmetric. The longest path has a length of 4 and the shortest a length of 3 so the diamond has an asymmetry of 1. Diamond 2 is symmetric.

4.2 Scanning procedure

To get results which are as close as possible to the actual usage across the entire Internet, they choose different locations as their sources for their scans which are spread as widely as possible around the world. Due to the non-availability of nodes in some regions and restrictions on tracing arbitrary devices from some nodes [5], it is not possible to reach an ideal distribution.

Due to the huge size of the IP address space and the costly scanning procedure, it is not feasible to scan routes towards all possible addresses. Because of that a preselection of destination addresses has to be done. Different approaches for generating destination lists have been used. For example a list of the addresses of the 500 most popular websites was used. Alternatively a list was used which was created by MIT researchers by running classical traceroutes to addresses from BGP tables of one of their routers and added the last responding host to the list [6].

For deeper analysis of the context in which the discovered load balancers are used, more information about their surrounding is needed. For example it is possible to analyze in which autonomous systems (AS) load balancers are used in. To find the AS for a discovered node IP to AS mapping services can be used [19].

4.3 Results

In this section, we will look at the results of two research groups. The first one around Augustin [6, 5] analyzed the usage of load balancers in the IPv4 internet. The second one around Almeida [2] analyzed the usage in the IPv6 internet.

4.3.1 Prevalence of load balancer types

	Overall			Filtered	
	Fraction of Balancers	% Routes IPv6	% Routes IPv4	Fraction of Balancers	% Routes IPv6
Per-destination	29.3%	43.5%	78.0%	29.2%	11.1%
Per-flow	50.0%	30.0%	54.8%	50.1%	17.7%
Per-packet	10.7%	30.1%	1.0%	10.6%	7.7%
Per-flow with TC	3.2%	14.8%	—	3.2%	3.3%
Per-application	6.0%	5.1%	—	6.0%	3.3%
Others	0.8%	1.2%	—	0.9%	0.6%
Total	100%	74%	92%	100%	29%

Figure 8: Overview of the prevalence of load balancers in IPv4 [6] and IPv6 [2]

Figure 8 shows an overview over their results about the prevalence of the different types of load balancers. The results Almeida et. al. received from their measurements showed very different results depending from the source of the scans. When analyzing their results they noticed that the sources which showed a very high prevalence of load balancers had a balancer one or two hops away from the source so almost all traces traverse this balancer. When ignoring these balancers all their sources showed very similar results. The values in the filtered column have these balancers removed if they are in the same AS as the source. [2]

doi: 10.2313/NET-2017-09-1_03

Augustin et. al. found out that per-destination load balancing is the most prevalent type in the IPv4 internet followed by per-flow load balancing. They found out that per-packet load balancing is used very rarely and seems to disappear. This can be explained with the negative effects of per-packet load balancing on TCP connections due to a high risk of packet reordering [7, 6].

Almeida et. al. also discovered per-destination, per-flow and per-packet, which were already used with IPv4, as the most prevalent types. Surprisingly they found a significant higher amount of per-packet load balancing than Augustin et. al. found for IPv4. Additionally, they found per-flow balancers which utilize the traffic class field in addition to the source and the destination port. They also found a small amount of load balancers which only use the TCP ports for load balancing [2].

4.3.2 Diamond characteristics

This section studies the properties of the discovered diamonds measured with the metrics introduced in Section 4.1. The measurements of Augistin et al. and Almeida et. al. have shown very similar results. So the characteristics are similar for IPv4 and IPv6 load balancing.

Diamond length. Most diamonds are short. Augustin et. at. found out that depending on the load balancer type the amount of diamonds with length two is between 26% (per-destination) and 90% (per-packet) [5, 2].

Diamond width. Most diamonds are narrow. Most diamonds have a max-width of less than 5. Additionally, Augustin et. al. discovered a few very wide diamonds with a max-width of 16 which is the maximum value for some vendors [17]. Such balancers are for example used to bundle a lot of low capacity links [5, 2].

Diamond symmetry. Most diamonds are symmetric. If asymmetry is present it is usually very low. Almeida et. al. discovered that most of the asymmetric diamonds they found belong to per-destination load balancers, so the asymmetry has no negative effects [5, 2].

4.3.3 Intradomain and Interdomain load balancing

Augustin et. al. also looked at the relation between diamonds and ASes. They defined load balancing with diamonds which are entirely inside one single AS as intradomain load balancing. In contrast, they defined load balancing with diamonds spread across multiple ASes as interdomain load balancing. [6]

Even if these two types do forwarding the same way they differ when looking at the routing protocols. For intradomain routing multiple intradomain routes can be installed in the forwarding table due to the equal-cost multipath capabilities of common intradomain routing protocols such as IS-IS [8] and OSPF [10]. In contrast interdomain routing which is done with the internet's interdomain routing protocol BGP does not allow to install multiple routes. Despite this limitation some vendors like Cisco [13] and Juniper [16] have build multipath capabilities for BGP [12] into their routers.

Augustin et. al. found out that most diamonds are created by intradomain load balancing. They analyzed their measurements from one of their sources, which is located in Paris, in detail towards intra- and interdomain load balancing and found out that 86% of all per-flow load balancers fit into a single AS. Diamonds which cross two ASes were rare and diamonds which cross three ASes were extremely rare. They discovered that it seems that the BGP multipath capabilities are disabled in most core routers. [6]

5. CONCLUSION

We have seen why the classical traceroute fails to deliver usable results in networks with load balancers. In detail we have seen that the distribution of packets to different links can lead to classical traceroute to leave nodes undiscovered, to detects nodes multiple times and to show nonexistent links in its result. Because these effects make the resulte unreliable the need for an improved tool, a multipath detection algrithm, is there.

Then we looked at the process of constructing such a mulitpath detection algorithm. First, we looked at nessessary changes for the classial traceroute. Due to these chages header fields which are evealuated by load balancers stay uncanged and the required identifier for a probe is stored in unused fields. Then, we covered how this modified traceroute can be extendet to a tool to reveal the complete topology of a route instead of only a single path. Therefore the routers on the route are tested to be load balancers with a stocastic approach. If a router is identified as a load balancer the different pathes behind it are examined separately. Addionally the type of the load balancer is examined with additonal probes.

Finally, we have looked at measurements which have been done with multipath detection algorithms and their results about the overall usage of load balancers in the internet. We have seen that load balancing is very widely used nowadays and most load balancing topologies are short, small, symmetric and do not span across multiple ASes.

6. REFERENCES

[1] traceroute(8) - traceroute for linux. `http://man7.org/linux/man-pages/man8/traceroute.8.html`. Accessed: 2017-04-07.

[2] R. Almeida, O. Fonseca, E. Fazzion, D. Guedes, W. Meira, and Í. Cunha. *A Characterization of Load Balancing on the IPv6 Internet*, pages 242–254. Springer International Publishing, Cham, 2017.

[3] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme. Ipv6 flow label specification. RFC 6437, RFC Editor, November 2011.

[4] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 153–158, New York, NY, USA, 2006. ACM.

[5] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 149–160, New York,

NY, USA, 2007. ACM.

[6] B. Augustin, T. Friedman, and R. Teixeira. Measuring multipath routing in the internet. *IEEE/ACM Transactions on Networking*, 19(3):830–840, June 2011.

[7] E. Blanton and M. Allman. On making tcp more robust to packet reordering. *SIGCOMM Comput. Commun. Rev.*, 32(1):20–30, Jan. 2002.

[8] R. Callon. Use of osi is-is for routing in tcp/ip and dual environments. RFC 1195, RFC Editor, December 1990. http://www.rfc-editor.org/rfc/rfc1195.txt.

[9] S. E. Deering and R. M. Hinden. Internet protocol, version 6 (ipv6) specification. RFC 2460, RFC Editor, December 1998. http://www.rfc-editor.org/rfc/rfc2460.txt.

[10] J. Moy. Ospf version 2. STD 54, RFC Editor, April 1998. http://www.rfc-editor.org/rfc/rfc2328.txt.

[11] J. Postel. Internet control message protocol. STD 5, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc792.txt.

[12] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). RFC 4271, RFC Editor, January 2006. http://www.rfc-editor.org/rfc/rfc4271.txt.

[13] Cisco Systems, Inc. Bgp best path selection algorithm. http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html. Accessed: 2017-04-12.

[14] Cisco Systems, Inc. How does load balancing work? http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/5212-46.html. Accessed: 2017-04-10.

[15] Cisco Systems, Inc. *Configuring a Load-Balancing Scheme*, chapter 4, page 59. Cisco Systems, Inc., San Jose, CA 95134-1706, USA, 2015.

[16] Juniper Networks. Configuring bgp to select multiple bgp paths. http://www.juniper.net/techpubs/software/junos/junos94/swconfig-routing/configuring-bgp-to-select-multiple-bgp-paths.html. Accessed: 2017-04-12.

[17] Juniper Networks. Configuring load-balance per-packet action. http://www.juniper.net/techpubs/software/junos/junos70/swconfig70-policy/html/policy-actions-config11.html. Accessed: 2017-04-10.

[18] Paris Traceroute. Paris traceroute. https://paris-traceroute.net/. Accessed: 2017-04-11.

[19] Team Cymru Inc. Ip to asn mapping. http://www.team-cymru.org/IP-ASN-mapping.html. Accessed: 2017-04-11.

doi: 10.2313/NET-2017-09-1_03

Holt-Winters Traffic Prediction on Aggregated Flow Data

Helge Brügner
Advisor: Johannes Naab, Jochen Kögel
Seminar Innovative Internet Technologies and Mobile Communications SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: helge.bruegner@tum.de

ABSTRACT

This paper investigates how varying parameters for aggregation of flow data impact network traffic predictions based on the Holt-Winters filtering algorithm. Changes in aggregation level (minute to daily granularity) and aggregation functions (e.g., mean, percentiles) are considered as well as a change in training set length. These evaluations are based on real-world flow data collected in a large enterprise network. For this, the two traffic prediction use cases "anomaly detection" and "capacity planning" are considered. It is concluded that Holt-Winters in combination with small aggregation levels and the 90th percentile serving as aggregation function perform best for short-term prediction with focus on anomaly detection. For long-term prediction as a tool for capacity planning, it is demonstrated that Holt-Winters on daily granularity data on the given data set is not sufficient.

Keywords

flow data, traffic prediction, anomaly detection, capacity planning, exponential smoothing, Holt-Winters filtering

1. INTRODUCTION

Visibility in enterprise networks is crucial for success. Having insight into a network is important for multiple reasons:

- debugging and analysis in case of faulty behavior of applications and networks,
- retrospective analysis of anomalies in the network, such as attacks and outages, and
- detection of possible bottlenecks in the network for capacity planning.

While having the possibility to analyze past traffic is sufficient for the scenarios described above, enterprise network traffic is subject to certain patters. These patterns are usually seasonalities (daily, weekly, yearly) and underlying trends that can be leveraged to estimate the future behavior of the network traffic. Machine learning algorithms for time series analysis are able to capture these patterns through the fit of a function to the given data points by leveraging various parameter estimation procedures. The resulting model is then extrapolated to give estimations about future values in the series and can often also quantify the certainty of such predictions. Two main use cases for such prediction in enterprise network traffic were identified:

Anomaly detection allows detection of significant traffic changes that do not fit a previously learned pattern. Such anomalies can range from attacks to an extraordinary number of requests due to a special offer on the website.

Capacity planning focuses on forecasting the underlying trend of the network traffic to enable early identification of possible future capacity issues.

In the implementation, both use cases consist of three distinct steps. First, a model satisfying the use case requirements is fitted to a training data set. Second, a certain number of future data points are estimated using the resulting fit. Third, the prediction is compared to reference values to identify special cases for further investigation by network operators.

For both use cases, the number of predicted data points depend on the granularity of the data (i.e., the number of data points per period). Anomaly detection will require fine-grained data sets (minutes, hours) while for long-term prediction for capacity planning, coarse granularities (daily, weekly) suffice. However, the reference values required for comparison vary per use case. For anomaly detection, actual measured network traffic data forms the reference to identify large deviations from the prediction and potentially raise an alarm. Capacity planning would instead use a fixed threshold as reference that, when likely to be exceeded in the near future, could also generate a notification.

Since preserving fine-grained historic traffic data can be highly storage space-consuming, an alternate strategy to overwriting old records is to aggregate the traffic data with increasing age. A possible aggregation strategy could be to use a granularity of 1 min for the past 4 weeks, then aggregate the data in 5 min intervals for a further 2 months, and then preserve data in 1 h granularity for a full year. A history over past years could then be kept by aggregating in 24 h intervals for another 3 years. Multiple functions for aggregation exist, such as the mean, summation, percentiles, or minimum/maximum operators, and others.

It is unclear whether the use of aggregated training data in prediction algorithms can produce meaningful results and how changes in granularity and the use of different aggregation functions impact the forecast results. The focus of this work will be to study how different aggregation levels and -functions change the accuracy of forecasts made by the Holt-Winters filtering algorithm in context of the anomaly detection use case. Furthermore, the impact of the training

doi: 10.2313/NET-2017-09-1_04

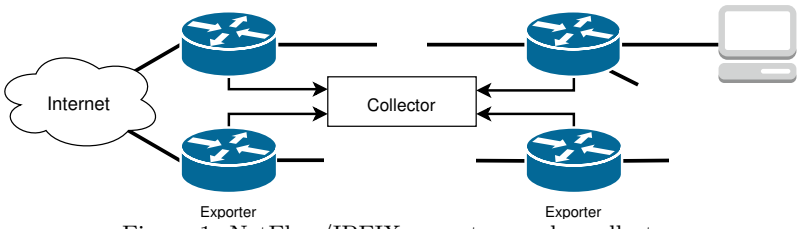


Figure 1: NetFlow/IPFIX exporters and a collector

set size to the forecast in capacity planning is investigated, again using Holt-Winters filtering. Two data sets exported from a large enterprise are used for the experiments; one with minute granularity and length of 4.5 weeks and a second set with daily granularity recorded over 3.5 years.

In section 2, this paper first describes two protocols for network traffic measurements. Then, the Holt-Winters filtering is introduced, followed by a comparison of quantification algorithms for prediction errors. After section 3 mentions related work, section 4 contains the analysis. First, in subsection 4.1, the test setups and the data sets used are described. Subsequently, subsection 4.2 compares aggregation levels, -functions, and -accuracies using the fine-grained data set with focus on the anomaly detection use case. The second part of the evaluation then analyzes different training set lengths and their impact on long-term traffic prediction accuracy for capacity planning. A conclusion in section 5 finally summarizes the results and presents possible future steps.

2. BACKGROUND

This section introduces two protocols commonly used in network traffic measurements, namely NetFlow and IP Flow Information Export (IPFIX). Then, both the additive and the multiplicative variant of Holt-Winters filtering are described. Finally, the most common algorithms to evaluate prediction errors are discussed.

2.1 Network Traffic Measurement

Network traffic measurements can be performed using either active or passive methods. Active methods such as Cisco IP SLA inject artificial traffic into the network by setting up special probes to actively measure the network performance. In contrast, passive methods monitor the existent traffic on key components in the network (e.g., on routers) by inspecting the transported packets. Since active probing requires changes in the network setup and anomaly detection as well as capacity planning potentially requires information about the packets flown, this paper will make use of passive network measurements.[5]

In contrast to the Simple Network Management Protocol (SNMP) that both allows to monitor and dynamically configure network devices[12], NetFlow and its successor IP Flow Information Export (IPFIX) focus on the monitoring of network devices (also referred to as "observation points") via "flows"[3]. Routers or switches usually function as such observation points. The corresponding RFC defines a flow as "set of packets or frames passing an Observation Point in the network during a certain time interval". Flows are identified by the "flow key", a user-defined set of packet properties used by the observation point to group encountered packets. Such a flow key can be, but is not limited to, the "5-tuple" consisting of source IP and port, destination IP and port, and protocol.[1][7]

The devices that host the observation points are called "exporters". They periodically send the packets containing flow information to "collectors". Collectors can potentially collect flows from multiple exporters simultaneously and perform pre-processing and aggregation for downstream analysis systems. A possible setup of exporters and collectors is shown in Figure 1.

Flows are closed and exported after either one of two specified timeouts occur. One of these timeouts is specified in the active case (i.e., packets of a certain flow are still arriving); the second occurs if the given flow has been inactive/idle for a certain amount of time (i.e., packets of a that flow were not seen during that time span). During these timeout intervals, the exporting devices caches packets and aggregates their relevant properties into a flow. Since NetFlow/IPFIX is capable of inspecting packets the observation points encounter, flow exports can also include information about packet content.[14] Exported values of observed packets can therefore be, but are not limited to, the number of bytes received, packet counters, or protocols observed, and range from OSI layer 2 (Data Link) to 7 (Application).[4]

Since SNMP is restricted to only exporting externally observable information about the encountered packets[12] (also referred to as the "interface view"), it cannot provide insight into the network apart from quantitative information regarding the packets encountered. Therefore, using NetFlow/IPFIX as a method to inspect the network flow is more promising since this potentially enables fine-grained analysis with focus on single hosts, subnets, or protocols.

2.2 Holt-Winters Exponential Smoothing

Time series analysis and forecasting can be performed using numerous different algorithms depending on the properties of the series. The network traffic data investigated in this paper shows seasonality (a pattern that repeats after a fixed number of iterations) while partially also exhibiting a trend over time. An filtering method that can incorporate both seasonality and trend is Holt-Winters filtering and belongs to the group of exponential smoothing procedures.

Exponential smoothing procedures are forecast algorithms that rely on updating equations to calculate predictions. In principle, exponential smoothing forecasts are "weighted averages of past observations, with the weights decaying exponentially as the observations get older"[9]. Multiple versions

doi: 10.2313/NET-2017-09-1_04

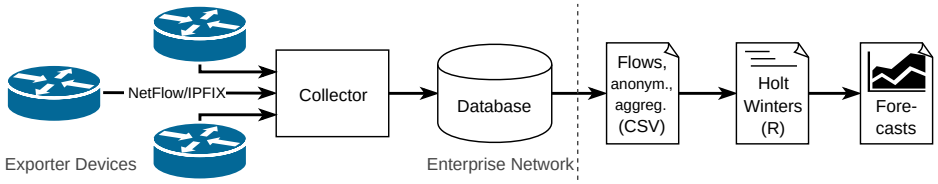


Figure 2: Test setup and evaluation work flow

Granularity	Obs./Week	Reasons
1 min	10080	Most common granularity for NetFlow/IPFIX
5 min	2016	Often used by network monitoring tools; originates from 5 min SNMP polling interval
1 h	168	Natural time interval
4 h	42	Divides 24 h granularity
24 h	7	Natural time interval

Table 1: The data set granularities and the respective number of observations per season (week)

exist, with the most basic form being simple exponential smoothing (SES). SES supports neither trend nor seasonality, but it forms the foundation for more sophisticated exponential smoothing models. Some of these models can support trend, some support seasonality, and some combine both.[2]

Another name for Holt-Winters filtering is "triple exponential smoothing" since it is based on three updating equations: ℓ_t models the level, b_t the trend, and s_t the seasonality of the time series. $\hat{y}_{t+h|t}$ represents the forecast at time $t+h$ given all the data points up to time t, and the constant m represents the seasonality (i.e., the number of observations per season). The equations are defined recursively, allowing for an iterative calculation of the predictions.[2][9]

Two variants of the Holt-Winters algorithm exist depending on how the a change in mean relates to the seasonal effect. The "additive" method is used if the seasonal effect is constant in each season (i.e., a change in mean does not impact the amplitude of the seasonal curve). Alternatively, the "multiplicative" covers the case when the seasonal effect is proportional to a change in the time series' mean. The additive updating equations are defined as follows:[2][9]

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t-m+h^+_m}\\ \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}\end{aligned}$$

In contrast, the multiplicative formulates the equations slightly differently:

$$\begin{aligned}\hat{y}_{t+h|t} &= (\ell_t + hb_t)s_{t-m+h^+_m}\\ \ell_t &= \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma\frac{y_t}{\ell_{t-1} + b_{t-1}} + (1-\gamma)s_{t-m}\end{aligned}$$

The equations for level, trend, and seasonality depend on three smoothing parameters α, β, and γ, respectively. To find appropriate values for the parameters, the most common approach is to first define an error term $e_t = y_t - \hat{y}_{t|t}$ with y_t as the training data point at time t and $\hat{y}_{t|t}$ as the estimated value of the algorithm. A minimization of the sum-of-squared-errors term $\sum e_t^2$ can then be used to estimate the three parameters.[2]

For the prediction of network flow data, Holt-Winters filtering seems reasonable since it provides a sufficient formulation of a model with support for both trend and seasonality as exhibited by the measured flow data.

2.3 Prediction Error Quantification

To evaluate the performance of forecasts it is necessary to quantify their error (i.e., the difference between predicted and actual values). Such calculations can either be performed in-sample or using a test set. In-sample error computation compares the fitted model to its training data, while out-of-sample error computation uses a subset of the whole data set for training and then compares the model's predicted values with the expected values from the test set. The former tends to support overfitting (i.e., a lack of generalization by specializing on the training data), which is why this section will focus on out-of-sample error computation.[9]

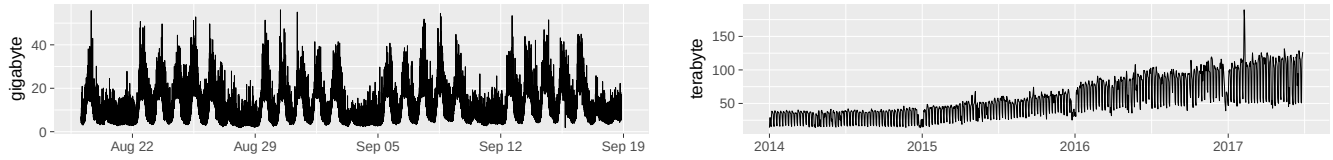
If different models are compared against the same data set, simple scale-dependent errors are often sufficient, while inter-data set comparisons require scale-independent error calculations to achieve comparability.[9] An example of inter-data set comparison is prediction of traffic for an entire network and comparing it with another prediction for a subnet of that network – the scale might differ by orders of magnitude. Since aggregation of the data set might potentially change the data's scale, the evaluations will use scale-independent errors for the calculation.

Scale-dependent errors calculate the error between predicted and actual values by subtraction: $e_i = y_i - \hat{y}_i$. If the scale of the test data is changed, e_i changes, too. Therefore, scale-independent percentage-based errors remove the scale by dividing by y_i: $p_i = 100^{e_i}/_{y_i}$. A popular variant of such errors is the mean absolute percentage error (MAPE):[9]

$$E_{\text{MAPE}} = \text{mean}(|p_i|)$$

Although more sophisticated versions such as the symmetric MAPE exist, Hyndman and Koehler [10] suggest not to use percentage-based scale-independent errors due to their as-

doi: 10.2313/NET-2017-09-1_04



(a) The short data set (b) The long data set

Figure 3: Plots of the original data sets

#	Date Range	Gran.	Periods	Data Pts.	Comments
1	19.08.2016, 13:00 – 18.09.2016, 20:14	1 min	≈ 4 (weeks)	44355	clear weekly seasonality visible, minimal trend, small amount of outliers
2	01.01.2014 00:00 – 29.06.2017, 01:00	1 min	≈ 3.5 (years)	1278	data aggregated by collector, damped trend visible, relatively clean data except a single large anomaly in 2017, 365 days per period except in 2016 due to the leap day

Table 2: The originally exported data sets used to evaluate the aggregation

sumption of meaningful zeros (i.e., a value of 0 indicates that the quantity measured is absent) as well as possible unstable calculations (e.g., an assumption must be that $y_i \neq 0$).[9]

Another group of accuracy measures not based on such assumptions are the "scaled" scale-independent errors. These make use of a scaling factor calculated using a naïve forecast on the training data set. Such scaling ensures that the result does not depend on the scale of data sets and can therefore also be used for inter-data set comparisons. Three variants to calculate such errors exist and can be used interchangeably depending on what patterns are found in the data. For data exhibiting seasonality, Hyndman [10] suggests using the seasonal naïve forecast ($q_{\mathrm{j,seasonal}}$) by assuming that current observations in the time series have the same value as the respective data point in the previous period. The mean absolute scaled error (MASE) is then defined by computing the mean over all errors:

$$q_{\mathrm{j,seasonal}} = \frac{e_j}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|}$$

$$\mathrm{MASE} = \mathrm{mean}(|q_j|)$$

If MASE = 1, the model used to compute the forecast is equally as good as the naïve forecast model used to compute the scaling factor. If the value is > 1, the forecast model performs worse, and consequently, if the result is < 1, it performs better.[10]

Due to the seasonality of the data and the recommendation to not use scale-independent, percentage-based errors, this paper uses MASE in combination with a the seasonal naïve forecast to compute prediction errors. The data sets are therefore divided into training and test set for fitting and evaluation, respectively.

3. RELATED WORK

In [6], Hellerstein et al. identified and inspected the same use cases (prediction for capacity planning and anomaly detection) for traffic prediction on a single web server. They base their calculations on the number of HTTP operations per second aggregated in 5 minute intervals in a data set with 8 months length. Based on this data, they construct a model from ground up that resembles the anomaly-free case and incorporates time-of-day, day-of-week, and monthly seasonality. By then analyzing the remaining error after applying the model to measured data, they propose an algorithm to detect "change points" that, as they argue, indicate anomalies. Additionally, they successfully 6apply the model to capacity planning by predicting the web server traffic multiple months into the future.

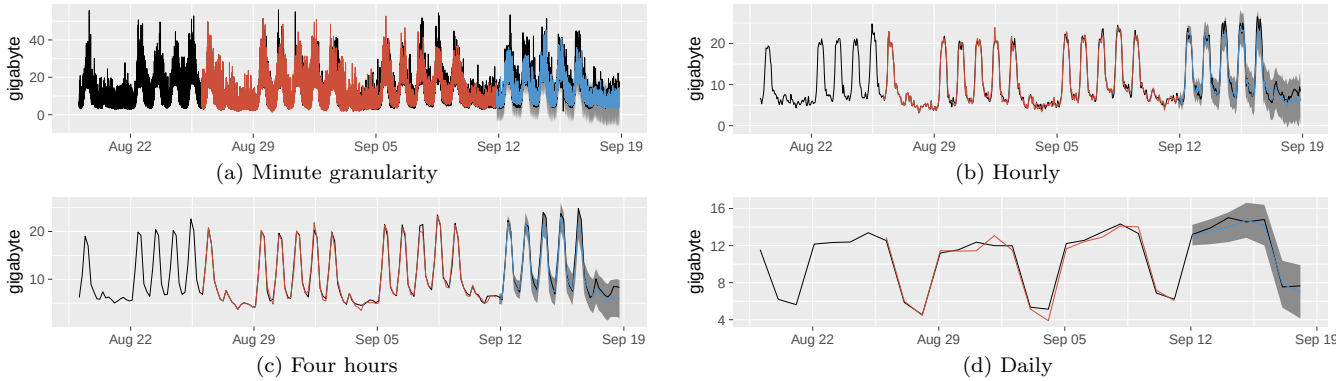
Münz [13] primarily focuses on anomaly detection in networks based on flow-level measurement data collected via NetFlow/IPFIX. Instead of only using bytes per flow, he incorporates additional metrics that are derived from the flow data for detection by using an approach called "multi-metric analysis". For the analysis, he compares multiple techniques for the detection, such as exponential smoothing, control charts, and principal component analysis. In addition to the pure detection of anomalies, he also describes automatic procedures to classify detected anomalies by relevance as well as to identify their causes. He concludes that the success of anomaly detection is highly dependent on the selection of metrics for classification.

In [15], Taylor and Letham propose a "scalable", "intuitive", "fast", and "accurate" way for forecasting business time series. For this, they make use of a model with components for growth, seasonality, and holidays. If the learning process is fed with information about, for example, holidays, the algorithm can learn the impact these days have on the time series, and consequently produce more accurate forecasts if such days occur again. They then developed a framework ("Facebook Prophet"[1]) that implements the algorithm they describe in R and Python. During the research, the author of this paper attempted to test the framework using a minute-granularity data set with around 44000 data points. The framework was unable to predict these points, and contact with the developers yielded that this was both due to the fine granularity as well as the resulting large data set[2].

In his work for the social media company Twitter, Kerjawal [11] developed the "AnomalyDetection" R package[3] to detect outliers in time series that describe the usage of Twitter's so-

[1] `https://github.com/facebookincubator/prophet/`
[2] `https://github.com/facebookincubator/prophet/issues/215`
[3] `https://github.com/twitter/AnomalyDetection`

doi: 10.2313/NET-2017-09-1_04



(a) Minute granularity

(b) Hourly

(c) Four hours

(d) Daily

Figure 4: Comparison of 1 min, 1 h, 24 h granularities aggregated using MEAN (the plot's scales differ)

cial media platform. The main use cases are the detection of rises in social engagement (e.g., during sporting events) and also identify malicious activity and its causes (e.g., spammers or bots). Instead of using fitting, the framework implements a method called "seasonal hybrid extreme studentized deviate" testing. This algorithm first applies decomposition of the time series into, for example, trend and seasonality, and then test if any of the residuals indicate anomalies. Due to its focus on anomaly detection, however, the framework cannot be used for prediction of the time series data.

4. EVALUATION

This section first introduces the test setup and the two data sets used. Then, it is investigated how a change in parameters impacts the traffic predictions produced with Holt-Winters. First, different aggregation levels are compared. Second, it is shown how aggregation functions impact the forecasts. Third, accuracies of forecasts with varying aggregation level and -function are compared. Finally, it is shown how a change in training set length impacts long-term forecasts.

4.1 Test Setup & Data Sets

Figure 2 displays the process from the data set acquisition to the generation of the test results. The exporter devices on the left are shown qualitatively; the actual number of monitored devices is approximately 50. The NetFlow/IPFIX exports are collected by a single collector instance that logs and pre-processes the flows and loads them into a database system. In the production environment, an analysis system would directly connect to this database and extract the data required for network analysis. However, in the test environment, the data is exported as CSV files including a timestamp and the total bytes flown in the network. This helps to make the tests reproducible and eases both anonymization and aggregation of the flow data. Subsequently, the data sets are fed into the Holt-Winters implementation built into the language R. The resulting model is then used to predict and plot the forecasts. Although an alternative implementation to Holt-Winters in R exists with the "ets" function from the package "forecast", the basic "HoltWinters"-implementation was preferred since ets does not allow for forecasting periods longer than 24 data points[8].

The data sets used are listed in Table 2. Set 1 was recorded in August and September 2016 over a period of a month in a network of a large enterprise with a minute granularity. The traffic of all NetFlow/IPFIX exporters in the network was aggregated by timestamp using a summation to represent the total number of bytes flown in the network at a certain point in time. Furthermore, the set only contains a small number of anomalies (e.g., in the form of holidays). A plot of the data set is shown in Figure 3a.

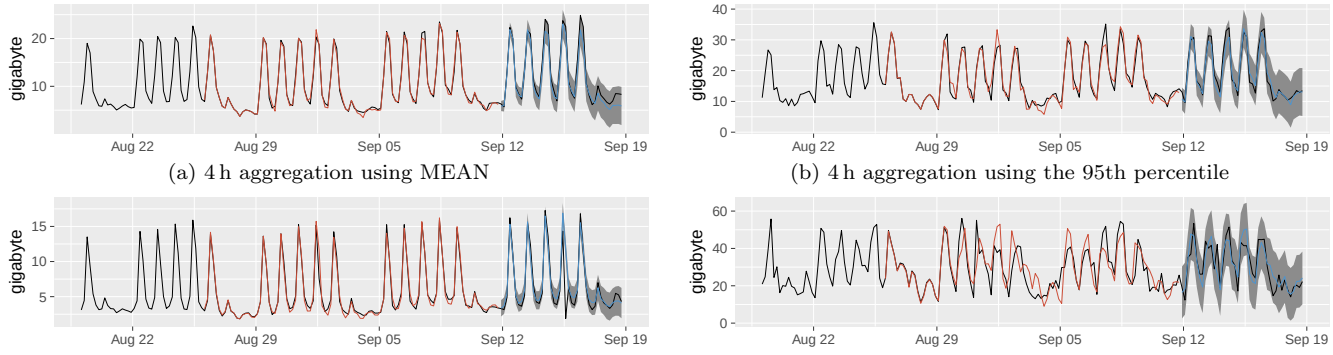
Data set 2 was aggregated by the collector prior to exporting it from the enterprise network to a granularity of 24 h using the MEAN function. Such aggregation is necessary due to storage constraints in the collecting system – a data set length of 3.5 years in a finer granularity was not possible while the data set was collected. As shown in Figure 3b, the data set contains a trend and both weekly and yearly seasonality. Furthermore, the seasonal effect is proportional to the time series' mean. A single large outlier is visible in the beginning of 2017, and furthermore, the data set experiences irregular seasonality due to the leap day added in the 3rd period (the year 2016).

The different levels of aggregation are produced by first dividing the minute granularity data into subsets with the size of the given aggregation factor f and then applying the aggregation function. Since the data sets are not aligned with the beginning of a day, the aggregation starts with the first observation and then aggregates the first f data points. This may result in an incomplete last aggregation, but this effect is minimal due to the large number of data points in general. The granularities investigated are listed in Table 1.

Seven aggregation functions are compared during the evaluation. MEAN and SUM are interesting since they both incorporate all data points from the respective aggregation interval, while MEDIAN (equal to the 50th percentile) has the interesting property of not being influenced by outliers. The 80th and 95th percentile have the same property as the MEDIAN and are also popular measures used for data sets. The two functions MIN and MAX are investigated as well since they could potentially reveal interesting properties about bottoms and peaks of the data set.

MASE is the measure used to evaluate the prediction accuracy. Since MASE requires a training set to compute the scaling factor and a test set to compute the actual accuracy, the data sets are split into such sets. In data set 1, the given

doi: 10.2313/NET-2017-09-1_04



(a) 4 h aggregation using MEAN

(b) 4 h aggregation using the 95th percentile

(c) 4 h aggregation using MIN

(d) 4 h aggregation using MAX

Figure 5: Comparison of MEAN, 95th percentile, MIN, and MAX aggregation on 4 h granularity data (the plot's scales differ)

4.5 periods (weeks) are splitted into approximately 3.5 periods for training and one period for testing, which comes close to the popular split of 80 % to 20 % for training and testing, respectively. Furthermore, the models are trained and evaluated on the same granularity. If, for example, hourly aggregation is used in the test set, hourly aggregation is also used in the training set.

4.2 Test Results & Comparison

With the data sets and the process described above, it is possible to test numerous combinations of aggregation levels and -functions to determine the best fit for the given use cases. The first three comparisons are based on the minute granularity data (data set 1) since this is the most useful for aggregation, and is therefore focused on the anomaly detection use case. Subsequently, it is investigated how Holt-Winters predictions and accuracies vary based on the size of the training set. Since it is useful to have a data set for such variations over a long period of time, these comparisons are based on data set 2.

The prediction plots all follow the same color scheme. The black line resembles the collected/aggregated flow data, while red and blue show the Holt-Winters fit and prediction, respectively. The gray area surrounding the blue prediction line is the 95 % confidence interval.

4.2.1 Aggregation Levels

Since it is of interest how the different levels of aggregation impact the ability of the Holt-Winters algorithm to predict the flow data, different aggregation levels are compared first. The graph shown in Figure 4a displays the data that has been exported with minute granularity originally which was then fitted with a additive Holt-Winters model. The black line shows that due to the fine granularity of a minute, a lot of noise is visible. This also appears to impact the training phase of the model, since both fit and prediction are noisy. Such a fit is likely the result of overfitting.

The data that was used to generate Figure 4b is based on the same data set as the previous plot but is aggregated with factor 60 (1 h granularity) using the MEAN function. The noise mostly is compensated by the aggregation function, which results in a less noisy fit. Besides preventing overfits, the noise reduction has another interesting effect. At noon and midnight (i.e., the peaks and bottoms of the fit), small drops and increases in capacity used become visible that were invisible in the 1 min granularity data. The increase at noon is likely caused by the employee's lunch breaks while the increase at night could be the result of scheduled nightly backups. If aggregated with 4 h granularity, the noise is further reduced, as visible in Figure 4c.

The graph shown in Figure 4d is plotted using flow data aggregated to 24 h granularity by again making use of the MEAN function. The noise that is still visible in the 1 h aggregation mostly disappeared, but as expected, any subday effects disappear as well. This, however, reveals different interesting aspects in the data. First, it can be seen that during the week, a peak in the traffic either appears during Thursday or Wednesday. Furthermore, the plot indicates that the peak day seems to change every other week. Furthermore, due to the lack of noise, the Holt-Winters fit appears to fit the data well.

4.2.2 Aggregation Functions

Figure 5 shows a comparison of four aggregation functions (MEAN, 95th percentile, MIN, MAX) on the same aggregation level, namely 4 h. The plots in Figure 4c and Figure 5a are both based on 4 h aggregation with MEAN for comparison.

As expected, the aggregations based on MEAN and SUM only differ in the scale. Both functions can therefore be used interchangeably depending on whether it is required to preserve the original scale (i.e., bytes per minute with MEAN) or the total number of bytes flown in each time interval (e.g., bytes per 4 h with SUM. Furthermore, the plots show that SUM, MEAN, MEDIAN and the percentiles preserve the seasonality well by either compensating (SUM, MEAN) or ignoring outliers (MEDIAN, percentiles).

Aggregation with the MIN function as shown in Figure 5c appears to also preserve seasonality at 1 h and 4 h granularity, because the data set appears to have a clear lower bound in the number of bytes tracked. For 5 min granularity a clear lower bound does not exist. Similarly, predictions based on 24 h MIN aggregation do not perform well since only the bottoms at night are preserved. Both result in a bad MASE value as shown in Figure 3. However, with 4 h granularity, MIN can reveal drops in the bytes captured, as seen on Thursday of the fourth week in the plot. The MAX func-

doi: 10.2313/NET-2017-09-1_04

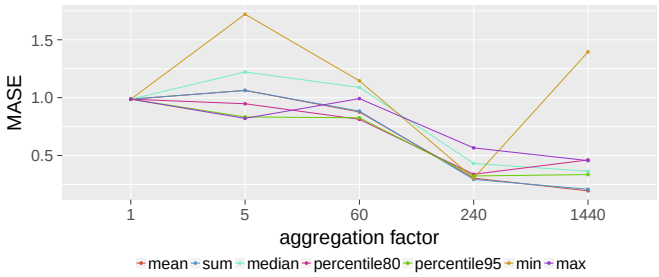
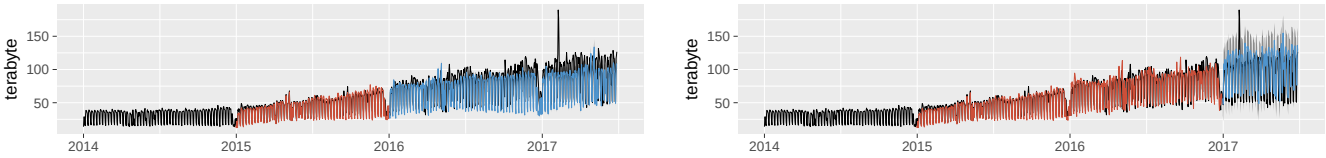


Figure 6: Comparison of prediction accuracy using different aggregation levels

	MEAN	SUM	MEDIAN	80th PT.	90th PT.	MIN	MAX
1 min	0.98	0.98	0.98	0.98	0.98	0.98	0.98
5 min	1.06	1.06	1.22	0.95	0.86	1.72	0.82
1 h	0.88	0.88	1.09	0.81	0.73	1.15	0.99
4 h	0.30	0.29	0.43	0.34	0.31	0.30	0.57
24 h	0.19	0.21	0.36	0.46	0.24	1.40	0.46

Table 3: MASE by aggregation levels and -functions



(a) 2 years training, MASE = 2.05

(b) 3 years training, MASE = 1.60

Figure 7: Comparison of training set lengths and the corresponding error

tion is shown in Figure 5d. In contrast to MIN, MAX does not preserve structure at any aggregation level due to the apparent lack of a clear upper bound in traffic.

4.2.3 *Aggregation Accuracies*

Figure 6 and Table 3 display the MASE of the Holt-Winters with varying aggregation levels and -functions. Since an aggregation factor of one results in the same data set for all aggregation functions, all plots start at the same error value for 1 min granularity.

The error for 1 min aggregation is insignificantly smaller than one. This is a sign of overfitting – the noisy fit of Holt-Winters in Figure 4a supports this hypothesis. Furthermore, the very similar error of the MEAN and SUM aggregation functions reveal that both perform almost equally, which is again due to the similarity of both aggregation function. Therefore, both are overlapping in the graph (here, the MEAN plot is covered by SUM). Aggregation using the 90th percentile performs most stable and results in MASE values greater than one6 for all granularities. The performance of the 80th percentile is close, which means that both functions ignore outliers well. With an increasing aggregation factor, outliers are more likely to be compensated, thus allowing MEAN and SUM to outperform the quantiles.

In general, a clear trend towards a better fit with an increasing aggregation level is visible. This can be explained by the decreasing number of data points and the resulting, increasing compensation of outliers, which prevents Holt-Winters from fitting the error and therefore modeling the data better. One clear exception is the MIN function – its plot shows outliers with large jumps and no clear trend. Although the plot of the MAX function accuracy does not show similar large jumps or poor MASE results, plots of its fit (e.g., Figure 5d) exhibit large confidence intervals in the prediction due to the lack of a clear structure in the aggregated data.

4.2.4 *Training Set Size*

The previous analyses primarily focused on short-term prediction due to the structure of the data set used. Data set 2, however, suits well for long term predictions due to its length of 3.5 years. The original data set is plotted in Figure 3b. Due to the length of the data set, a comparison of fit accuracies with varying training set lengths (2, 2.5, 3 years) is possible. Such a comparison is crucial, since it can reveal for what period of time a forecast can be trusted. Furthermore, the multiplicative formulation of Holt-Winters is used since it allows to capture the multiplicative effect in the data.

The resulting forecasts with their MASE result are shown in Figure 7. Looking at Figure 7a shows that the fit can estimate the data relatively well until mid-2016 and then clearly underestimates due to the growing trend in late 2016. Similar results with a MASE value of 2.02 have been collected with a training set length of 2.5 years, however, the plot has been omitted for simplicity. In Figure 7b, the fit overestimates the actual trend immediately.

The fits for 2 and 2.5 years of training have in common that the prediction is off towards the end of 2016 due to the additional leap day. This effect is not visible in the third plot since the beginning of Christmas 2016 is still part of the training set. However, in the last fit, the irregular seasonality learned might impact the prediction at the end of 2017.

In general, the poor MASE result of ≥ 1.60 for all training set lengths indicates that the prediction based on Holt-Winters was unable to learn the pattern and trends of this data set sufficiently.

5. CONCLUSION

In conclusion, short-term prediction for use case 1 appears to be promising. The best function for the given data is the 95th percentile-function since it results in good MASE results for all aggregation factors. MEDIAN, MIN, and MAX should generally be avoided unless special requirements have

doi: 10.2313/NET-2017-09-1_04

to be met. Provided that the data contains a seasonal, stable lower bound, MIN could, for example, be used to detect anomalies resulting in bandwidth drops (e.g., a failure of a core router) as a scenario of the anomaly detection use case.

Given that the proper aggregation function is used, short-term prediction appears to work well for small aggregation levels (potentially between 5 min-1 h). Aggregation generally seems to be a powerful tool to compensate for outliers to prevent overfitting and, therefore, enable better detection of anomalies. Anything with a coarser granularity than approximately 1 h should, however, not be used for anomaly detection since this potentially results in a loss of important sub-day effects. In such a case it may become impossible to detect anomalies either due to the delay in aggregation or suppression of such outliers by the aggregation function.

For long-term prediction required for the capacity planning use case, the best aggregation function still remains unclear due to the lack of data. The MAX function could potentially be a good candidate since it may preserve peaks crucial for capacity planning. Daily granularity appears to be sufficient, since it preserves weekly seasonalities and reduces the number of data points in the data set to a size suitable for fast learning.

It also remains unclear which prediction algorithm works best for long-term prediction. Due to the poor MASE results and the lacking possibility to incorporate special days as well as seasons with different lengths, Holt-Winters may not be a good candidate in this scenario – although this claim should be verified by testing further data sets. Simpler algorithms such as a simple fitting with a trend component only or more sophisticated frameworks such as Facebook Prophet may constitute alternatives and could be subject to further investigation.

Future work could potentially also investigate more sophisticated aggregation functions, such as an aggregation by business hours (i.e., only use values between 6AM and 6PM for aggregation). Additionally, it may be interesting to analyze subsets of the data (e.g., specific interfaces, protocols). Furthermore, a better training could result in even better fits with Holt-Winters. A possibility for that might be to train the model with a coarse-grained training set ($\geq$1 h) and evaluating it against a fine-grained test set ($\leq$5 min).

Acknowledgement

This work was partly performed at IsarNet Software Solutions GmbH and funded as part of the AutoMon project by the German Federal Ministry of Education and Research (BMBF) with contract number 16KIS0408K.

References

[1] P. Aitken, B. Claise, and B. Trammell. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC 7011. 2013. DOI: 10.17487/RFC7011. URL: https://rfc-editor.org/rfc/rfc7011.txt.

[2] C. Chatfield. *The Analysis of Time Series: An Introduction*. 6th ed. Texts in statistical science. Boca Raton, Fla. and London: Chapman & Hall/CRC, 2004. ISBN: 9781584883173.

[3] B. Claise. *Cisco Systems NetFlow Services Export Version 9*. RFC 3954. 2004. DOI: 10.17487/RFC3954. URL: https://rfc-editor.org/rfc/rfc3954.txt.

[4] B. Claise, P. Aitken, and N. Ben-Dvora. *Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX)*. RFC 6759. 2012. DOI: 10.17487/RFC6759. URL: https://rfc-editor.org/rfc/rfc6759.txt.

[5] L. Cottrell. *Passive vs. Active Monitoring. Stanford Linear Accelerator Center*. Stanford Linear Accelerator Center, 2001. URL: https://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html (accessed on 08/13/2017).

[6] J. Hellerstein, F. Zhang, and P. Shahabuddin. "Characterizing Normal Operation of a Web Server: Application to Workload Forecasting and Problem Detection". In: *In Proceedings of the Computer Measurement Group*. Morgan Kaufmann, 1998, pp. 54–61.

[7] R. Hofstede et al. "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[8] R. Hyndman. *Forecasting with long seasonal periods*. 2010. URL: https://robjhyndman.com/hyndsight/longseasonality/ (accessed on 09/29/2010).

[9] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and practice*. Heathmont: OTexts, 2016. ISBN: 978-0987507105. URL: https://www.otexts.org/fpp (accessed on 06/20/2017).

[10] R. J. Hyndman and A. B. Koehler. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2006.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0169207006000239.

[11] A. Kejariwal. *Introducing practical and robust anomaly detection in a time series. Twitter Blog*. Twitter Blog, 2015. URL: https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html (accessed on 06/01/2017).

[12] D. R. Mauro and K. J. Schmidt. *Essential SNMP*. 2nd ed. Sebastopol, Calif. and Farnham: O'Reilly, 2005. ISBN: 978-0-596-00840-6.

[13] G. Münz. *Traffic Anomaly Detection and Cause Identification Using Flow-Level Measurements*. Vol. 2010,06. Network architectures and services. München: Network Architectures and Services Techn. Univ. München, 2010. ISBN: 3937201122.

[14] G. Sadasivan et al. *Architecture for IP Flow Information Export*. RFC 5470. 2009. DOI: 10.17487/RFC5470. URL: https://rfc-editor.org/rfc/rfc5470.txt.

[15] S. J. Taylor and B. Letham. *Forecasting at Scale. Facebook Research*. 2017. URL: https://facebookincubator.github.io/prophet/static/prophet_paper_20170113.pdf (accessed on 08/13/2017).

Using the blockchain to add automated financial incentives to the Public Key Infrastructure

Justus Fries
Advisor: Heiko Niedermayer
Seminar Innovative Internet-Technologien und Mobilkommunikation SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: fries@in.tum.de

ABSTRACT

The Transport Layer Security Protocol in its current implementation is based on a centralized and intransparent infrastructure. These flaws have been the cause of Man-in-the-middle (MitM) attacks, which are most commonly rooted in compromised Certificate Authorities. Log-based enhancements, such as Certificate Transparency, have made an effort to solve these problems by logging every signed certificate and thus making signing a public process. However these systems lack proper financial incentives and automation. In this seminar paper Instant Karma PKI (IKP) [15], a system to improve on Log-based enhancements and on the Public Key Infrastructure (PKI), is described and discussed.

Keywords

IKP,PKI,certificate,transparency,blockchain,incentives

1. INTRODUCTION

Secure data transfer on the internet is based on the Transport Layer Security protocol (TLS) [8]. It allows two actors on the web to communicate in privacy and provides data integrity. The most commonly used protocol based on TLS is HTTPS [18], which allows transfer between client and server in an end-to-end encrypted connection.

The basis for the trust model in TLS is the Public Key Infrastructure (PKI). The PKI has two main actors, the websites that want to establish trust in their public key for secure communication, and the Certificate Authorities (CA), who sell a certification of those public keys. The trust in those certificates is established via a chain of trust, which is rooted in certificates bundled with web browsers and operating systems. An example for this can be seen in figure 1. The main point of failure in this system are the CAs themselves. Worldwide, CAs have been compromised and used for Man-in-the-middle attacks. Examples for this are TURKTRUST [10] from Turkey, VeriSign [16, 23] and Comodo [19] from the U.S., DigiNotar [2] and GlobalSign [24] from the Netherlands. One of the newest incidents came from Symantec [21], one of the largest CAs. The issue with these compromises is that every single CA can issue certificates for any website and thus compromise even big websites like Google or Facebook. This problem is facilitated by the fact that browsers contain hundreds of root certificates.

One approach to solve this, proposed by Google themselves, called Certificate Transparency (CT) was published as an experimental RFC in 2013 [14]. It is a log-based approach that is supposed to be publicly monitored, thus making detection of unauthorized certificates faster and easier.

The authors of IKP [15] identified two main goals with their system. It should add incentives for all actors to behave correctly and everything should be automated. This will be described in more detail in Section 3.1. In Section 2 background information to understand IKP is presented, Section 3 deals with the components of the Instant Karma PKI and evaluates its functionality.

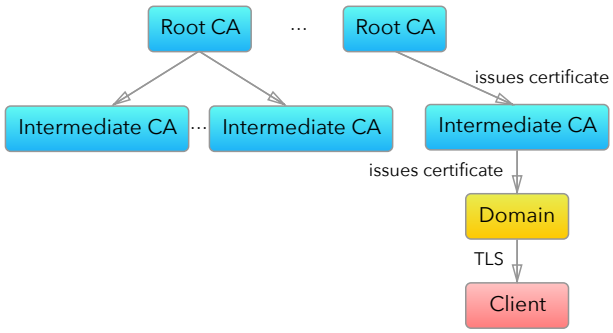


Figure 1: Chain of trust in the public key infrastructure.

2. BACKGROUND

This section describes background information that is important to understand the IKP architecture and why its implementation is desirable.

2.1 Log-based enhancements

Log-based enhancements such as Certificate Transparency [1] aim to move the process of signing certificates to the public and thus make detecting malicious certficates faster by keeping a log of every certificate signed. This log has certain cryptographic features, for example being append-only, that allow verification of the logs themselves and of the existence of a certificate in the log. Furthermore certain important characteristics such as invulnerability to malicious actors [9] can be proved. In the CT system the browsers leverage these features to extend the TLS protocol to include verification that a certificate has been logged using the Signed Certificate Timestamp extension to X.509 certificates [7], this verification of logging is part of the extended validation process. Log operators are for example Google or CAs. Other important actors are the monitors who watch for suspicious

doi: 10.2313/NET-2017-09-1_05

certificates in the logs and reporting those. They also make sure that all logged certificates are visible by periodically fetching new entries and exchanging this information with other monitors and auditors using a gossip protocol.

2.2 Ethereum and smart contracts

Ethereum [26] is a blockchain-based cryptocurrency. Similar to other such currencies it relies on its decentralized structure to reach a consensus on a public ledger containing blocks of transactions.
The feature that makes Ethereum suitable for implementing IKP are smart contracts. While originally defined as a digital version of contracts, Ethereum takes them one step further and allows executing arbitrary code in contracts which means the consensus also has to enforce correct contract execution. It achieves this by introducing a new type of account, the contracts, which are created by sending a transaction from an externally controlled account to the empty account with the code to be executed as data. While smart contracts can contain programs and state they can only be executed or react to something if a transaction is sent to the contract account, thus making code execution in the system dependent on actions of externally controlled accounts. Computation power, which is essentially a state change of the contract, is paid for in a seperate currency from Ethereums currency, called gas. The price for gas is relatively low, however each contract has to have a self imposed gas limit.
However smart contracts are not fool proof, can have bugs and can be attacked [3]. An example of this is the DAO hack [17], where a large contract had severe problems that caused unwanted money transfers. In the end this lead to a fork of the blockchain, which is generally speaking undesirable because it disrupts the value of the currency and reduces trust in the consensus mechanism.

3. INSTANT KARMA PKI

This section is going to describe the Instant Karma PKI [15] system with all its components and afterwards discuss and evaluate it.

3.1 Analysis

To describe a system such as IKP it is important to keep in mind what problems exist and how it attempts to approach them, the authors of IKP identified the following problems [15]. Certificate Transparency, as briefly described in Section 1, doesn't solve important problems that exist in todays PKI infrastructure.
One flaw of CT specifically is that the list of authorized logs is centralized, thus adding another point of failure, the list of trusted logs shipped with browsers. Other log-based enhancements have tried to solve this [12], however none have been implemented, so actual security of alternatives is hard to verify.
Another obvious problem is running logs and monitors. Running logs is expensive for actors not participating in the PKI, specifically anyone but CAs, and some logs in the current CT system are run by CAs who were compelled by Google to do so due to security incidents [20]. Furthermore running monitors is best done by domains themselves since they know best what authorizes a certificate for their domain, while this requires additional setup, services like SSLMate's Cert Spotter, which is available on github [22], make this easier even for smaller actors.
An important issue is the difficulty of actually reporting unauthorized certificates. Only the CAs themselves can currently revoke certificates within a reasonable timeframe and reporting to them means manual effort for the detector. Other possibilities for the detector would be contacting browser vendors and getting the root certificate revoked, however rolling out browser updates takes testing and time and thus won't be finished in a short amount of time either. An example for where this works is the repeated misbehavior of Symantec, who Google decided to take action against by severely limiting Symantecs root certificates in Chrome [21].
Another observable issue is that CAs need to invest more in their security and their verification process. As seen in Section 1 a lot of CAs have been the target of hackers who then proceeded to execute Man-in-the-middle (MitM) attacks, thus increasing their security is the obvious move. Some CAs also seem to take the verification of ownership very lightly and gave intermediate certificates to companies that don't enforce any verification [13]. They need to be given incentives to invest in their security, and the verification of domains and of companies they sell certificates to.
With these problems in mind, IKP's goals are to add financial incentives for CAs to behave correctly, besides the monetization of certificates, to automate verification of suspicious certificates and to reward the detectors accordingly and to achieve this in a decentralized way [15].

3.2 Overview

This section defines the main goals as identified in the IKP paper [15]. The main idea of IKP is to automate a decentralized system that creates financial incentives for handling CA misbehavior. To do so it is important to define CA misbehavior, so it has to introduce a concept that allows the definition of authorized certificates. To also add automated financial incentives for reporting certificates, IKP needs the ability to evaluate those certificates and it has to provide a way to respond in case the evaluation results in the identification of CA misbehavior.
These ideas result in a system that should fulfill the following properties:

- Auditability of the information in the IKP system that define authorized certificates.
- Automation of reactions to CA misbehavior without any third parties
- Financial incentives that ensure actors who show good behavior get rewarded
- Punishment of CAs for misbehavior resulting in discouragement of more misbehavior

Before going into details, actors in this system are established and possible adversarial behavior of those actors is described. The three obvious actors, are the ones that interact in the TLS protocol's system, namely domains, CAs and clients. Another important actor is the detector, who reports suspicious certificates. Since IKP is based on the idea of adding financial incentives to the PKI, the goal of all actors is a positive Return of Investment (ROI). For CAs

generally speaking this means issuing an unauthorized certificate but not being penalized for it or receiving more payments than penalties. Domains can act malicious via collusion attacks together with detectors and CAs, and detectors can try to report every certificate they come across and report it, hoping for one of them to actually be unauthorized and thus receiving rewards.
In TLS the domains buy certifications of their public keys in order to verify their identity to clients during the TLS handshake and establish a secure connection for transferring data. In order to achieve IKP without impact on the existing infrastructure, a new entity is introduced: the IKP authority. As seen in Figure 2 it enables domains to register so called Domain Certificate Policies (DCP), which are a concept for describing CA misbehavior. DCPs allow the domain to define what makes a certificates authorized in a way that can be checked by the IKP authority automatically. CAs are enabled to register Reaction Policies (RP) to the IKP authority. These policies are bought by domains from CAs and are supposed to be the financial incentive for CAs to not issue unauthorized certificates, because if no unauthorized certificates are issued against the domain until the expiry date, the CA is rewarded. They act as an insurance for the domain against misissuance, as the domain and the detector receive rewards from the RP if an unauthorized certificate for it is detected. The IKP authority also executes the important inspection of the certificates that detectors report using the DCP that the domain published and issues payouts based on the RPs. Furthermore the authority controls a global fund to send and receive payments, as a result it has the role of a trustee to all other parties.
To register in this system all entities have to first register in the Ethereum blockchain. This is necessary in order to receive payments and also to interact with the IKP authority, since it is implemented as a smart contract and thus can only be interacted with by sending transactions to it. This is what results in a fully automated system and enables IKP to run with little supervision. Another result of the authority being a contract is that the essential part of DCPs and RPs have to be implemented as a contract too, such that the IKP authority can automatically execute them in case it has to check a reported certificate or send payments from the reaction policy.
To summarize, the IKP authority has to store DCPs and RPs in order to automate misbehavior checking, the CAs issue RPs to domains, the domains publish DCPs to articulate what authorizes a certificate for them and the detectors report suspicious certificates, thus starting the checker functionality of the IKP. The authority contract has to escrow funds and send out payments according to the results of checks of certificates.

3.3 IKP authority

This section further describes the functionality offered by the IKP authority as it is described in the IKP paper [15]. Similar to having their root certificate in web browsers, **CAs** need a way to **register** in IKP if they want to sell insurance policies. To start this they have to register an externally controlled account in Ethereum and then send a transaction to the IKP contract with the fields depicted in Figure 3. The *CA name* identifies it, the *Valid From* field specifies at what point the registration information is valid, the *payment account* is the address of the CAs Ethereum account,

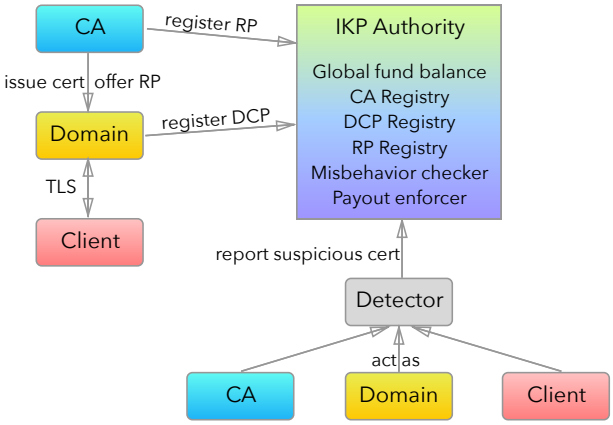


Figure 2: Overview of IKP showing its entities and corresponding functions. Adapted from [15].

note that it does not necessarily have to be the account the registration transaction came from. The *public keys* are the CAs signing keys that are contained in their root certificate, the *update keys* are an optional security measure that allows CAs to define other keys in case their primary account is compromised and lastly the *update threshold* defines how many update key signatures are needed to update the registration information. It is important to see that while the update mechanic adds a certain amount of security, it is not a complete protection against compromise, private keys still have to be kept secret. However since CAs already have to store their certificate signing keys in a secure way, securely storing update keys and the Ethereum key should not be a problem.

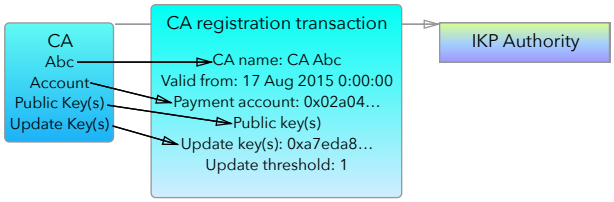


Figure 3: Overview of CA registration in IKP. Adapted from [15].

Domains have to be able to **register** in the IKP system too, since they have to issue the DCPs which are crucial to the whole system. The idea is to use DCPs not only to describe authorized certificates but also to act as a way of registering. The domain has to register similar information to CAs, this is described in detail in Section 3.4.
The IKP also offers a fair exchange mechanism for payment transfers during **RP issuance**. The procedure is as follows: First the domain sends the payment for the price/fee for the RP and a hash over the RP to the IKP authority, then the CA creates and sends the RP to the authority. The authority then verifies that the sent RP corresponds to what the domain is expecting to buy from the CA and that certain payout constraints are fulfilled. If anything goes wrong all payments are returned and the process is cancelled. If everything is correct the IKP authority will send the price the domain paid for the RP to the CA and all funds in the

doi: 10.2313/NET-2017-09-1_05

transaction that created the RP are transferred to a global fund that the IKP authority maintains.
With this, after the domain and the CA negotiated the contents of the RP, the IKP acts as an escrow service for the associated payments. This escrow functionality can also be used to issue certificates, since both parties who participate in certificate issuance, the CA and the domain, already manage Ethereum accounts to participate in the IKP system, it makes sense to offer this service as the IKP authority itself is an easily auditable smart contract.
Furthermore the IKP has to accept **reports of misbehavior** from detectors. This usually happens in the form of sending a suspicious certificate and Ethereum account information. The detector has to pay a fee for reporting certificates in order to discourage sending every single certificate a detector encounters and hoping one of them is unauthorized.
Another important part of this is a so called pre-report containing the reporting fee and a hash of the certificate and a secret. This is enforced so that blockchain miners cannot replace the detectors name with their own and thus receive the rewards for detecting a malicious certificate. This is called commitment hash and the commitment is opened after a certain number of blocks are mined by sending the actual certificate and the secret. Note that the reward for a detector is supposed to be negotiated between domain and CA when buying a RP and furthermore has to be larger than the initial reporting fee. The reward is only sent out if the IKP authority detected misbehavior.
If misbehavior is detected, not only is the detector paid, the reaction program also contains **payouts** to the domain and the CA. The financial transactions are executed by the IKP authority. The penalty in the CA occurs in the form of less payout than it would have received without misissuance.

3.4 Domain Certificate Policies

This section describes the design goals for DCPs, the actual implementation and functionalities that the checker program has to offer, as described in the IKP paper [15]. The last paragraphs give information about the functionalities the IKP authority has to provide and leverage in order for DCPs to function.
Domain Certificate Policies are an important aspect of the IKP architecture since they allow domains to publish what properties an authorized certificate has to fulfill, which then defines CA misbehavior. Following are the design principles the IKP creators had in mind when developing DCPs.
Firstly the information contained in DCPs that allows checking a certificate and results in the assessment of the certificate, is only determinable by domains themselves. Thus IKP allows only domains themselves to specify what authorizes a certificate for them. As a result it is possible to move away from terminology established in CT, the concept of a suspicious certificate, and concretely define authorization criteria. This also enables detectors who know these criteria to report unauthorized certificates with corroboration from domains themselves.
Secondly to enable detectors to determine criteria for authorized criteria, DCPs are stored in a decentralized way. This is enabled through the blockchain and enables the information to be globally consistent through the consensus mechanism. This also allows the authenticity to be affirmed by anyone, thus making determining authorization of certificates globally consistent.
Lastly the criteria to determine authorization have to be as expressive and detailed as possible. This allows for varied policies and this is important since larger organizations probably have more complex requirements for their certificates. This is realized by implementing the checker program as smart contracts that returns a boolean value describing authorization, `true` if a certificate is authorized, `false` otherwise. An interesting observation is that while programs can be arbitrary since Ethereum offers turing-complete languages to program contracts, the gas cost for certain operations is relatively high and reaches the upper limit of compute power contracts are allowed to use. An example for this would be parsing X.509 certificates [7], the type of certificate used in TLS. This resulted in the current implementation of IKP having to employ different parsing techniques. However since the method provided by the checker contracts does not change the program's state, they only have an initial gas cost for deploying them and afterwards are free to use by the IKP authority to check whether a specific certificate is authorized or not.

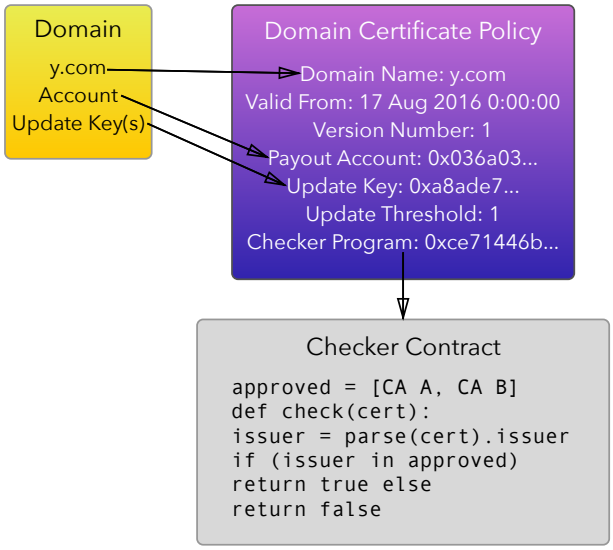


Figure 4: Overview of a Domain Certificate Policy. Adapted from [15].

DCPs themselves, or rather the transaction that the domain has to send, have to contain information about the policy itself, such as *Valid from* and *Version number* fields, but also information about the domain, since DCPs are to also function as a domain registration as described earlier. All fields that have to be part of the DCP can be seen in Figure 4. Information for domain identification are the domain name, payment account and update key fields. The *Domain name* is the domain's DNS name, the *Payout account* is the domain's Ethereum account it intends to send and receive payments in the IKP system from and the *Update keys* allow the domain to recover from a compromised or lost payment account. The *Update threshold* defines how many signatures of update keys are needed to update information in that case and lastly the *Checker program* binds a specific checker contract to the DCP. This checker contract has to provide a method `check()` that the IKP authority can call when checking a specific certificate. Identifying the policy is

doi: 10.2313/NET-2017-09-1_05

implemented via the *Valid from* field, which specifies when the DCPs comes into effect, the *Version number* field which identifies the new DCP, but only if the checker program is replaced or modified, and the checker program, which contains the account key of the checker program contract. The *Valid from* and *Version number* fields are further used to bind a specific checker program to a RP, since running every single DCP the domain published, in order to verify a reported certificate, is time intensive and could lead to conflicting information. The specific interactions are described in Section 3.5.
To realize the functionality of DCPs the IKP authority has to be able to offer a way to prove ownership of a domains during registration. To achieve this either the existing DNS or TLS infrastructures are leveraged. The domain signs its name and DCP either with its DNSSEC private key and includes a signature chain to the DNS root or in the case of TLS it signs with its TLS private key and includes a chain to a root CA. The first option is the preferred way of handling this since the DNS root has less of a history of security incidents. However, according to measurements by the IKP authors [15], DNSSEC is not widely adopted. Thus the TLS approach is the more realistic one, however it requires the IKP to have a list of accepted root CAs just like browsers. Thus it is important to keep the amount of trusted CAs minimal, so that DCPs cannot be registered with rogue certificates. For updates the IKP authority has to verify signatures signed with the domain's update keys and ensure the minimum amount specified by the domain is met.

3.4.1 *Examples for DCP checker contract functionality*

This section will describe possible functionality of checker contracts and operational functionality the IKP authority has to offer to run the system. As defined earlier, the policies, implemented as checker programs, should be expressive. The following example implementations can be combined with each other to achieve a set of desired criteria for a domain's certificates. Since contracts can call each other's check method, if a set of criteria is already implemented by another domain they can be included in the domain's own contract using boolean relations such as AND or OR.
The standard for certificates in TLS is X.509, so the fields available in such a certificate are provided as parameters through parsing. The first obvious kind of program are whitelists for CAs. The contract takes the *Issuer Name* field and matches it against a list of authorized CAs. Another whitelist can be implemented by checking the key identifier contained in the certificate. Similar to public key pinning, where domains specify a list of trustable keys, the *Subject Public Key Info* field can be extracted and matched against a list of authorized and pinned keys. This is a different approach compared to current implementations of certificate pinning [11] because of the decentralized storage of the list. Short-lived and wildcard certificate rules are also supported by extracting the *Not Before* and *Not After* fields to determine the lifetime of a certificate and extracting *Subject Name* to make sure * does not appear. Wildcard certificates specifically enable MitM attacks on all subdomains, since wildcard certificates are valid for every single subdomain. As in CT's implementation it is possible to enforce that a certificate has been logged, similar to the signed certificate timestamp (SCT).

3.5 Reaction Policies

This section explains the goals for designing RPs, their implementation and lastly gives an overview over the different scenarios that involve CA misbehavior or good behavior and how the financial rewards work in these situations, as described in the IKP paper [15].
Unlike Domain Certificate Policies, Reaction Policies only serve a single purpose, which is acting as an insurance for the domains against misbehavior. However just like with DCPs there are three main design goals. They are intended to prevent unwanted incentives and consequences.
Firstly RPs should be independent from certificates, because while they are both sold by CAs to domains, they are supposed to be an addition to certificates and are supposed to protect domains against misbehaving CAs, so binding them to one specific certificate doesn't make sense.
Secondly RPs are issued for a specific DCP, specifically they are bound to a single version of a DCP, which means there is exactly one checker program the IKP authority has to check against if a certificate issued by the CA that issued the RP is reported. This means the domain has to be registered via a DCP in the IKP system before being able to purchase RPs.
Thirdly a RP is only valid against a single occurrence of CA misbehavior. This is mainly done because the IKP authority has to make sure there are enough funds available to issue payouts. A domain can however hold multiple reaction policies, so the only limiting factor is the transaction that has to take place in order to register the RP in the IKP authority.

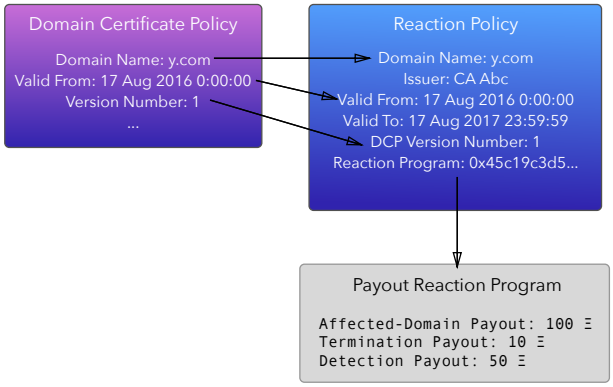


Figure 5: Overview of a Reaction Policy. Adapted from [15].

As with DCPs, RPs have to be sent as a transaction to the IKP authority. Figure 5 shows the different fields the policy has to include. The *Domain name* field, which identifies who the RP was issued to, the *Valid from* field, which describes the time from which the RP becomes active and the *DCP version number*, which binds the triggering of an RP to a specific checker contract, all serve the same purpose as in DCPs and bind a reaction policy to specific domain policy. The *Issuer* field serves as an identification of the CA that issued the policy, the *Valid to* field which denotes when the RP runs out without any occurrence of misbehavior and lastly the *Reaction program* contains the address to a reaction contract account. The reaction program, similar to the checker program, has to define specific methods. The `trig-`

`ger()` method is called when an unauthorized certificate is reported via the IKP system for the domain specified in the policy. If that is the case, the IKP authority also records the date when misbehavior occurred. The `terminate()` method is called by a domain in case a CA issued an unauthorized certificate and the domain wants to terminate the RP early, because it lost trust in the CA. In this scenario the IKP authority has to compare the *Valid from* field with the date of misbehavior occurrence. The `expire()` method is what ends the contract at the date specified in the *Valid to* field and is called by the CA that issued the RP.
As seen in Figure 2 the IKP authority acts as a registry for RPs. This is implemented as a mapping of domains and a list of their active RPs. This list is ordered by the *Valid to* field. This ensures that the RP that expires the soonest is triggered first.

3.5.1 Payouts and constraints

The reaction program in Figure 5 is just a sample of what can happen as a reaction. While the program has to provide three methods, it is not defined what they do, which differentiates it from checker programs that have to return a boolean value. However the overall goal of IKP, as defined in Section 3.2, is creating financial incentives, so the only contract discussed here and the original paper [15], are payout reaction programs. The goal was to keep this as generic as possible and thus specific payment amounts are not enforced or given.
The **affected-domain payout** is the amount the domain receives if any CA that is registered in IKP issues an unauthorized certificate for the domain. This payment does naturally not occur if the CA is not registered with the IKP authority. It is supposed to compensate the domain for the risk of MitM attacks.
The **termination payout** is a payment that gets split between the CA that issued the RP and the domain that bought it. This payment occurs whenever the reaction contract is terminated early, there can be multiple reasons for this, which will be explained in Section 3.5.2. The payment is split proportionally to the amount of time left in the RP's validity period in such a way that the domain receives less money the longer the RP is active. The amount the domain receives is bound by a minimum that is defined globally in IKP and the total amount defined in the payout contract.
The **detection payout** is the financial incentive for the detectors for monitoring logs and CA operations and for reporting unauthorized certificates they may find. Note that this payment is only made if the reported CA is registered in the IKP system and as such can be higher than the initial fee a detector has to pay for reporting since the amount can be imposed on the misbehaving CA. This also means if the CA is not registered in IKP the detector simply gets the initial fee paid back.
To issue an RP the domain and the CA negotiate the terms of it outside the IKP system. If the RP is negotiated as a payout contract, they negotiate the price/fee and just described payouts. There are two constraints that must be held up: The affected-domain payout plus the minimum termination payout for the domain must be larger than the price. This is a measure against collusion attacks between domains and CAs or domains and detectors and guarantees a negative ROI if all payouts are summed up. The price must be larger than the termination payout itself, so that an issuing CA can still profit from a RP that is terminated, when the issuing CA wasn't the one that misbehaved. As mentioned in Section 3.3 the detection payout must be larger than the fee the detector has to pay when reporting certificates.

3.5.2 Scenarios of (mis)behavior

This next section lays out possible scenarios that describe reactions to CA misbehavior and correct behavior and how aforementioned payouts are involved, as described in the IKP paper [15].
If a CA issued an RP for a domain and there are no occurrences or detections of misbehavior by this CA during the validity time, the RP simply expires and the issuing CA receives the money it escrowed by the IKP authority for the CA. This escrowed funds are not defined in the final version of the IKP paper [15], however in the first version it is described as a fraction, between zero and one, of the sum over affected-domain, termination, detection and global fund payouts.
If a domain terminates an RP early, for example if the issuing CA misbehaved against someone else, it is paid its fraction of the termination payout and the CA is paid the escrowed funds minus the domains termination payout.
For the next part the concept of internal and external misbehavior is introduced. Since the IKP system can only penalize CAs that are registered with it, it distinguishes between internal, where the offending CA is registered with the IKP authority and external misbehavior, where the CA is not part of IKP.
In the case of internal misbehavior we distinguish between the CA that misbehaved and the CA that issued the RP since the RPs are ordered by validity ending time, and thus the case, where the misbehaving and the issuing CA are the same entity, is rare. The detector sends the reporting fee to the IKP authority, which then detects the misbehavior. It then makes the misbehaving CA pay the affected-domain payout and the detection payout to its global fund. Afterwards it pays the domain the affected-domain payout and its part of the termination payout, which will be relatively close to the minimum payout that is globally defined, since the oldest contracts are triggered first. The detector receives the detection payout and the CA that issued the RP receives all escrowed funds minus the domain's fraction of the termination payout.
For external misbehavior penalization of the CA that misbehaved is impossible. Thus the detector only gets its reporting fee back and no payout. The domain again receives a fraction of the termination payout and the RP issuing CA receives the escrowed funds minus the domain's split.
Lastly if an RP expires, which means the time specified in the *Valid to* field is in the past and no misbehavior happened, the IKP authority removes the RP from the mapping mentioned earlier and sends any payments made by the CA during issuance, which were escrowed, back to it.

3.6 Evaluation

After describing IKP, this paper now evaluates IKP and describes possible problems and implementation challenges. It is important to note that while there seems to be a github for IKP, `https://github.com/syclops/ikp`, it is apparently not available to the public.

doi: 10.2313/NET-2017-09-1_05

3.6.1 Weaknesses of the architecture

Firstly the negotiation and effectiveness of RPs heavily depend on the size of the participating entities [15, p. 7]. One strategy that CAs specifically could employ is minimizing the detection payouts. While this wouldn't work when negotiating with big organizations wanting to participate in IKP, it certainly works with smaller actors. This could cause all RPs a domain owns to have low detection rewards and thus can remove financial incentives from reporting rogue certificates for all small domains.
Secondly CAs have to willingly participate in the IKP system in order for it to be effective. This means the financial incentives have to outweigh the risk the CAs incur by being openly penalized for misbehavior.
The system allows the fair exchange mechanism used for RP issuance to be used for certificate issuance too, however this adds more bloat to the system and does not seem to make sense financially since cryptocurrencies are rather unstable.
Lastly if for example a single person detects an unauthorized certificate, there's an initial hurdle for them by having to register in Ethereum in order to send transactions to the IKP authority. While this system is clearly enhancing financial incentives for log operators and monitors, it does not put a focus on single detections.

3.6.2 Problems and risks incurred by the implementation in Ethereum

Another risk for CAs are vulnerable contracts. As seen in the DAO hack [17], a contract cannot be updated easily if it is found to be vulnerable and thus a CA could potentially lose escrowed money. Another example for this is the Parity multisig bug, which recently occurred due to method passthrough functionality in Ethereum contracts [5].
The current version of IKP is also not implementable due to RSA signature verification [15, p. 12], which is presumably needed for update keys, not being a part of the current Ethereum implementation for smart contracts. It heavily reduces the cost from 3000 gas to 200. There exists a proposal on the Ethereum proposal github [4], however it has been open for over a year as of the writing of this seminar paper and noone is assigned to it.
Another problem that is based on Ethereum is that while smart contracts are written in a turing complete language and thus can produce arbitrary programs, there is a maximum limit on gas, which limits actors with complex checker contracts. Furthermore it can take a significant amount of effort to formulate a contract for large entities with complex requirements for certificate authorization.
An aspect that is important to consider is privacy. In a system such as IKP that enables an insurance system, transactional privacy is a desirable property. Ethereum lacks this feature, however the next major release of Ethereum, Metropolis, is supposed to include privacy enhancing features [6].
The recent development of a decompilation tool called Porosity [25] has shown that reverse engineering smart contracts is a focus in research. This can become a serious problem for the security of checker contracts, since they contain sensible information. This could for example allow an attacker to decompile a domain's checker contract and attack specific CAs listed in the contract, if it is implemented as a simple whitelist.

4. CONCLUSION

Instant Karma PKI is meant to add financial incentives to the PKI that disincentivize misbehavior and reward the absence of it. A blockchain is used to add natural financial incentives and keep the access to the whole system decentralized. The choice of Ethereum specifically allows for a large part of the system to be completely automated. In order to define misbehavior, it allows the only actor that truly knows when a certificate for itself is malicious, the domain, to define what authorizes certificates issued for it. While lacking in implementation, the system is well thought out and is relatively cheap to run due to low gas costs in Ethereum. This system does not solve the problem of misbehavior occurring itself, however doing so would probably require a full rework of the whole PKI and reworking systems on the internet is difficult and adoption for new standards is low, this can be seen with IPv6 for example. It also cannot function properly if no CA willingly joins the system, however the hope is that the financial incentives outweigh the risk of being penalized for misbehavior.

5. REFERENCES

[1] Certificate Transparency. https://www.certificate-transparency.org. Accessed: 2017-07-01.

[2] H. Adkins. An update on attempted man-in-the-middle attacks. https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html, August 2011. Accessed: 2017-07-01.

[3] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on Ethereum smart contracts. Cryptology ePrint Archive, Report 2016/1007, 2016. http://eprint.iacr.org/2016/1007.

[4] A. Beregszaszi. Support RSA signature verification. https://github.com/ethereum/EIPs/issues/74, March 2016. Accessed: 2017-07-01.

[5] L. Breidenbach, P. Daian, A. Juels, and E. G. Sirer. An In-Depth Look at the Parity Multisig Bug. http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/. Accessed: 2017-07-26.

[6] V. Buterin. Ethereum Research Update. https://blog.ethereum.org/2016/12/04/ethereum-research-update/. Accessed: 2017-07-27.

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFC 6818.

[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.

[9] B. Dowling, F. Günther, U. Herath, and D. Stebila. Secure Logging Schemes and Certificate Transparency. Cryptology ePrint Archive, Report 2016/452, 2016. http://eprint.iacr.org/2016/452.

[10] P. Ducklin. The TURKTRUST SSL certificate fiasco - what really happened, and what happens next? https://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-

happened-and-what-happens-next/, January 2013. Accessed: 2017-07-01.

[11] C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP. RFC 7469 (Proposed Standard), Apr. 2015.

[12] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 679–690, New York, NY, USA, 2013. ACM.

[13] A. Langley. Maintaining digital certificate security. https://security.googleblog.com/2015/03/maintaining-digital-certificate-security.html, March 2015.

[14] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), June 2013.

[15] S. Matsumoto and R. M. Reischuk. IKP: Turning a PKI Around with Decentralized Automated Incentives. In *S&P'17: 38th IEEE Symposium on Security and Privacy (Oakland)*, 2017.

[16] J. Menn. Key Internet operator VeriSign hit by hackers. http://www.reuters.com/article/us-hacking-verisign-idUSTRE8110Z820120202, February 2012. Accessed: 2017-07-01.

[17] R. Price. Digital currency Ethereum is cratering because of a $50 million hack. http://www.businessinsider.de/dao-hacked-ethereum-crashing-in-value-tens-of-millions-allegedly-stolen-2016-6, June 2016. Accessed: 2017-07-01.

[18] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFCs 5785, 7230.

[19] P. Roberts. Phony SSL Certificates issued for Google, Yahoo, Skype, Others. https://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311/75061/, March 2011. Accessed: 2017-07-01.

[20] R. Sleevi. Sustaining Digital Certificate Security. https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html, October 2015. Accessed: 2017-07-01.

[21] R. Sleevi. Intent to Deprecate and Remove: Trust in existing Symantec-issued Certificates. https://groups.google.com/a/chromium.org/d/topic/blink-dev/eUAKwjihhBs/discussion, March 2017. Accessed: 2017-07-01.

[22] SSLMate. Cert Spotter. https://github.com/SSLMate/certspotter. Accessed: 2017-07-26.

[23] T. Sterling. Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard. https://technet.microsoft.com/library/security/ms01-017, March 2001. Accessed: 2017-07-01.

[24] T. Sterling. Second firm warns of concern after Dutch hack. https://www.yahoo.com/news/second-firm-warns-concern-dutch-hack-215940770.html, September 2011. Accessed: 2017-07-01.

[25] M. Suiche. Porosity: A Decompiler For Blockchain-Based Smart Contracts Bytecode. DEF CON 25, July 2017.

[26] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. Technical report, 2015.

An introduction to Public and Private Distributed Ledgers

Jan Felix Hoops
Advisor: Dr. Holger Kinkelin
Seminar Innovative Internet Technologies and Mobile Communications SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: jan.felix.hoops@in.tum.de

ABSTRACT

Blockchains are fairly novel systems that offer highly reliable, non-mutable and trustworthy storage of records. Applications include distributed payment systems like Bitcoin and smart contract systems like Ethereum. This paper first gives an introduction to blockchain technology using the example of Bitcoin. One of the foci is the so called consensus algorithm that controls which new block can be added to the blockchain. Furthermore we explain why the consensus algorithm of "traditional" blockchains is not suitable for smaller application scenarios such as corporate networks and also is highly wasteful concerning energy consumption. As a next step, the Linux FoundationâĂŹs Hyperledger Project is introduced, which is an umbrella for various open source projects concerning private blockchains. Here the focus is on the blockchain implementation Sawtooth, which features a new consensus algorithm called Proof of Elapsed Time that makes use of a specific hardware security feature of novel Intel CPUs. This algorithm features the best properties of Proof of Work while also eliminating the huge waste of power. However, the dependency on the specific CPU features can also be seen as a major weakness of the concept.

Keywords

blockchain, private blockchain, distributed ledger, consensus, Bitcoin, Hyperledger, Sawtooth, Proof of Work, Proof of Elapsed Time

1. INTRODUCTION

Blockchains are one of the most impactful technologies of the new millennium, even though much of their potential still remains undiscovered. They finally allow for the realization of truly distributed ledgers that do not require a trusted third party. They also raised the bar for non-mutability and reliability, thanks to their high redundancy. These properties make blockchains attractive for numerous public and corporate applications everywhere there are records to keep [12].

The technology emerged in 2008 when Satoshi Nakamoto presented Bitcoin [24], the first decentralized, digital currency. That is why it is the example used to illustrate the explanation of blockchains in section 2. To this day, public crypto-currencies are still the dominant applications using blockchains. More recent ones like Ethereum [5] do not only offer currency, but expand on the idea. Ethereum for example further innovates by featuring a smart contract system.

Even though private blockchains (addressed in section 3) have not seen as much success as public ones yet, there are a lot of big companies actively supporting ongoing research. This difference in development compared to their public counterpart is mainly caused by traditional consensus algorithms not scaling down well. It will be argued in section 3.2, that applying Proof of Work to a small network essentially breaks the system. However, progress is being made and one promising project is presented in section 4. The project in question is Intel's Sawtooth. It is a modular enterprise solution for building private blockchains. Sawtooth features some architectural upgrades compared to a traditional blockchain, but most importantly introduces a new consensus algorithm: Proof of Elapsed Time. This algorithm is very promising for future blockchain development, because it features the strengths of Proof of Work, while also eliminating the large amount of wasted energy.

2. PUBLIC BLOCKCHAINS

A blockchain can be seen as distributed database with high redundancy. It is replicated using a peer-2-peer system and every participating node keeps a full copy. The major benefit of a blockchain is that it provides high trustworthiness without relying on a trusted third party. The absence of a trusted third party involved in consensus also renders blockchains more secure by removing an angle of attack while also cutting costs and complexity.

The following overview will be using the example of Bitcoin. However, the text is kept as generic as possible. Bitcoin is a crypto-currency using a blockchain as a distributed, public ledger and probably the most commonly known application of blockchain technology. The focus of this overview is on the architecture of the blockchain, not the underlying network. Technical details presented in the following subsections were taken from the Bitcoin Developer Reference [2], that can also be recommended for information about the underlying peer-2-peer network.

2.1 Transactions

In contrast to a database, a blockchain stores no tables, but transactions. Generally a transaction transfers an asset from one individual to another. These assets can be of any imaginable kind and the individuals do not need to be human.

Bitcoin transactions are financial transactions. Their structure can be seen in Table 1. As indicated by the fields `tx_in count` and `tx_out count`, a transaction is not limited to only one sender or receiver. These fields contain the number of

doi: 10.2313/NET-2017-09-1_06

Table 1: Bitcoin Transaction Structure [2]

Name	Data Type
version	uint32_t
tx_in count	compactSize uint
tx_in	txIn
tx_out count	compactSize uint
tx_out	txOut
lock_time	uint32_t

Table 2: Bitcoin Block Structure [2]

Name	Data Type
fixed value	0xD9B4BEF9
block size	uint32_t
block header	header
transaction counter	varInt
transactions	transaction list

inputs and outputs (`tx_in` and `tx_out`) of a transaction. Every input references a previous output and contains proof of ownership. Every output specifies the amount of bitcoins to spent and the new owner. Ownership in Bitcoin is handled via asymmetric cryptography. Every individual is identified by a bitcoin address computed from the public key of it's asymmetric key pair.

Blockchain transactions can also be timed. Bitcoin implements this using the `lock_time` field. A transaction may not be processed before the time given there. The field's unsigned integer value is parsed dependent on size: below 500 million it is interpreted as a block height and above as a unix epoch.

In Bitcoin there is one special transaction: the coinbase transaction. It is used to create new bitcoins from nothing. Technically this is realized by using only one input, the coinbase input. It is like a normal input except that it has placeholder values for most fields. The necessity of this transaction is related to mining and will be discussed in section 2.3.

2.2 Blocks

Blocks are used to group up transactions. They consist of a header for meta data and a list of transactions. Every block is connected to its predecessor by containing the hash of the predecessor's header in its own header. The header

Table 3: Bitcoin Block Header Structure [2]

Name	Data Type
version	uint32_t
previous block header hash	char[32]
merkle root hash	char[32]
time	uint32_t
nBits	uint32_t
nonce	uint32_t

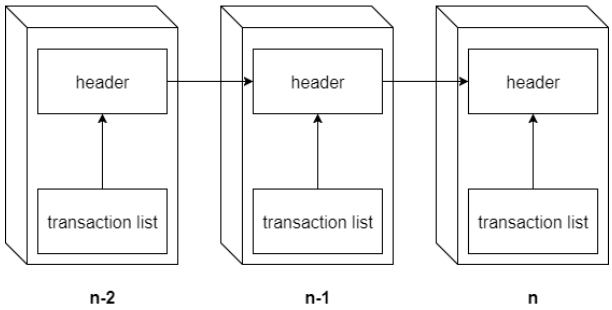


Figure 1: Simplified Blockchain Structure
Every arrow symbolizes a hashing operation. The origin is the value being hashed and the target is where the hash value will be included.

hash is used to uniquely identify a block. By constructing a merkle tree over all transactions of a block and storing the root in the header of said block, it is ensured that the transactions also influence the block header hash. This secures the transactions, because it allows for easy detection of manipulation.

Table 2 shows the structure of a block as used by Bitcoin and the corresponding header can be seen in Table 3.

Bitcoin implements the `previous block header hash` by hashing twice with `SHA256`. The `merkle root hash` is the hash in the root of the merkle tree hashing all transaction ids of this block. If however the block only contains one single transaction (coinbase transaction), its transaction id is used as the merkle root. The `time` field contains a time stamp in the standard unix format, making it possible to enforce the `lock_time` found in transactions. Both `nBits` and `nonce` field hold values important for Bitcoin's consensus mechanism, which will be addressed in section 2.3. In addition to all of these specifications, blocks can also have a data size limit. For Bitcoin, a block may never exceed 1MB in size.

Figure 1 shows the essence of a blockchain's structure. The schematic shows part of a blockchain with emphasis on the chain of hashes. Every arrow symbolizes a hashing operation. It can be seen, that the header of block `n` contains the hash of the header of block `n-1`. These are the horizontal arrows. The vertical ones show the transactions of a block being hashed into its header.

Only the very first block of a blockchain has to deviate from the exact shown structure as it has no predecessor. This block is a kind of dummy block. It is called the genesis block and contains placeholder values. This block is hardcoded in every client for a blockchain.

2.3 Consensus with Proof of Work

The basic problem a blockchain solves is distributed consensus. In a blockchain this means that all participating nodes have to agree on the newest block added to the chain, meaning they implicitly agree on one chain of blocks. In order to achieve consensus, some form of voting is necessary among these nodes. This consensus mechanism is the

doi: 10.2313/NET-2017-09-1_06

heart of every distributed ledger and arguably its most defining feature. The ledger's characteristics (e.g. transaction throughput, supported number of nodes) all depend mostly on this mechanism.

Bitcoin uses a leader election protocol (refer to section 3.2) in order to decide on a node being eligible to create the next block. This specific system is the *Proof of Work* (PoW) system. When creating a block, a *crypto-puzzle* has to be solved. To be considered valid, a block's header hash has to be less or equal to the current target threshold. This threshold is adjusted regularly in such a way that a new block is created every 10 minutes on average and is encoded in the `nBits` header field. The header hash is computed by hashing twice with `SHA256`.

A node trying to solve this crypto-puzzle and creating new blocks is referred to as a *miner*. Miners collect transactions requested by other nodes and group them up into a block. In order for the header's hash to meet the required threshold the miners are able to modify the header in the following ways: First and most obvious, they can vary the `nonce` field in the block header. Second, the order of the transactions may be varied (except the coinbase transaction) in order to change the `merkle root hash` in the header. Third and last option is to change one of the fields (`coinbase script`) in the coinbase input.

As long as the used cryptographic hashing function is secure, solving this puzzle is computationally expensive due to the miners simply having to brute force a solution. Because of this, this system can be viewed as a lottery with computational power only affecting the likelihood of winning.

A miner has to invest in hardware and electricity. In order to incentivize nodes to do mining, Bitcoin has a block reward. This is a certain amount of bitcoins that is created in the coinbase transaction as a reward to the miner in addition to the transaction fees. However this reward is halved every 210,000 blocks. That is necessary in order to limit the amount of bitcoins that will ever be created and in doing so, preserving their value. On the downside this means that in the not so distant future the transaction fees will be all the reward a miner receives.

2.4 Blockchain Security Mechanisms

Anyone can create a transaction, but creating a valid one requires proving the ownership of the asset(s) in transfer. Validity in this context means that other nodes of the network will consider the transaction fit to be included in a block on the basis of the current blockchain (e.g. asset was not already spent) and the common specification (i.e. correct transaction format). The transaction is then published onto the network where every node on the way will also check the transaction's validity before distributing it further. If found to be valid, a transaction will be kept by mining nodes to be incorporated into a block. Only after a transaction was published inside a block, it is considered to have been conducted.

Whenever a node solves the proof of work it has finished a new block. The validity of this block depends on the proof of work matching the current threshold, the `merkle tree root` matching the transactions as well as their order and the validity of each transaction. The node then distributes the new block along the network. Every node along the way checks the block's validity before distributing it further and adding it to it's local image of the blockchain.

Since a miner can append his new block anywhere in the chain, the blockchain may fork. Bitcoin's solution to this problem is simple yet effective: In this event the longest chain of blocks is the one agreed upon. This works because creating blocks is expensive. An individual will never have the computational power needed to fork the chain and create a longer chain than all of the other nodes in the meantime. Honest nodes will always switch to mining on the highest valid block, because they will not get any rewards if their mined block is not part of the longest chain. By working on a successor to a block, a node implicitly expresses it's acceptance.

For this blockchain to work there are mainly two assumptions made. The first one is that the majority of nodes are honest. The second one is that no node ever reaches more than 50% of the networks computational power. Both of these make sure that the majority of blocks is always created by honest nodes.

2.5 Attacks on Public Blockchains

This section introduces three attacks on public blockchains and evaluates them. The main purpose of this section is less to show weaknesses of public blockchains, but rather to illustrate why they are so secure.

The most realistic attack on bitcoin is the *double spend* [25]. The goal of this attack is to spend the same transaction output twice. This is possible to a certain extent because a block can later on become invalid if another branch of the chain becomes longer than it's own one. An attacker could buy an item from a seller. As soon as his transaction is encapsulated in a valid block, the seller would give the item to the attacker. Then the attacker would need to fork the blockchain right before his transaction and make it longer than the original branch. This way he could void his transaction while already having obtained the item. In reality this is very difficult to execute as it requires vast amounts of computational power and even then this is still a gamble. As the attacker would most likely have less than 50% of the networks computational power he would have to be lucky in order to overtake the original branch. To conclude, it can be said that this attack is theoretically possible, but not feasible due to the amount of funds and/or luck it requires.

Another take on the double spend attack is referred to as *fast double spend*. Some exchanges with bitcoin require a certain speed. A supermarket, for example, can not wait for about 10 minutes for the next block to encapsulate the transaction. These vendors instead rely on just seeing the customer's transaction on the network. An attacker could possibly create the anticipated transaction while also creating a malicious second transaction that spends the same transaction output as the first one, but this time transfers it to another wallet under the attacker's control. Then both transactions are distributed across the network. The attacker has to make sure the first transaction is broadcasted

to the vendor while broadcasting the malicious transaction to most other nodes. This increases the chances of the malicious request being mined first. As both transactions contradict each other, only the first one to be mined will ever be mined. For more detail on this attack refer to [21]. In practice it is still rather difficult and depends on luck.

All of these double spend attacks can be averted by simply waiting. With every block succeeding the one with the transaction in question it gets exponentially more difficult to undo the block. That is because an attacker would have to fork the chain in front of that block, redo all succeeding blocks and overtake the original branch all while the network is still mining on said original branch. After only about 6 succeeding blocks a Bitcoin transaction can be considered rather safe [25]. On the downside, this means having to wait for about an hour.

It should also be mentioned that another angle of attack is strategically withholding and publishing mined blocks such as described in [20]. This kind of attack is referred to as *selfish mining*. Normally an individual should earn a percentage of the total block rewards approximately matching it's percentage of the network's total computational power. The goal of this attack is to increase the percentage of the reward without increasing computational power. Although there are several theoretical strategies for attacking Bitcoin like that, there is no evidence of them ever having been executed.

2.6 Problems in Public Blockchains

A major drawback of the PoW system is the large amounts of computational power it requires. All of this energy is constantly used and is essentially wasted. The Bitcoin network is rapidly expanding and the continuous electricity consumption is projected to be in between the production of a power plant and the consumption of a small country like Denmark by 2020 [19]. But even if this waste is considered to be worth it, there has to be an incentive for the nodes to provide computational power. In a system like Bitcoin, that is only designed to be currency, this incentivization is easy, but there are numerous applications for blockchain that do not involve currency.

Another problem caused by the PoW system is the limited transaction throughput. As the system is tuned to produce only one block (on average) in a certain interval of time, this interval and the block size dictate the transaction throughput [18]. This is a hard maximum that can cause transactions to pile up and essentially clog the blockchain. This maximum can only be increased by changing the specification of the distributed system. That process however is very tedious as can be seen with Bitcoin's block size discussion currently going on [3].

The need to have a full copy of the blockchain for most applications leads to large amounts of data every node has to hold. Currently the complete Bitcoin blockchain already takes up more than 130 Gigabytes of disk space [4]. Even though this can be considered affordable with respect to today's hardware, it also creates an ever rising entry barrier, because every new node joining the network has to download all blocks created so far.

Because a node in a public blockchain may have a full copy of the current blockchain, it has full access to all transactions. For this reason it is impossible to conduct a transaction without everyone else knowing. Even worse: from the transactions the current balance of any individual can be computed. Privacy is not possible in a public blockchain.

3. PRIVATE BLOCKCHAINS

A private blockchain is a blockchain that restricts read and write access using additional security systems. This restriction can be achieved by an access restricted blockchain deployed on a public network, as well as a standard blockchain deployed on a private network. This normally results in a lot fewer nodes being part of the network and typically all of these nodes can be authenticated. This type of blockchain is also referred to as *permissioned* being the contrast of a public, or permissionless blockchain.

3.1 Motivation

The research in the field of private blockchains is motivated by the aforementioned problems found in public blockchains. Private blockchains are mainly envisioned to be used as secure ledgers in and in between companies, but the possibilities are endless.

Blockchains remove an angle of attack on the system by getting rid of an all-powerful admin. Furthermore a blockchain provides better auditability than most current systems. Last, but not least, a blockchain might also save costs, because depending on the implementation of the private blockchain, there is no or just limited involvement of a third party. These aspects are so impactful that blockchain has the potential to forever change a broad range of transaction-based industries [12].

Privacy is required for mainly one reason. It is necessary in order to maintain common business practices like private contracts. Doing so also protects company data and therefore is essential in ensuring competitiveness.

3.2 Consensus

In order to understand why consensus in private distributed ledgers is so challenging, some exposition on consensus mechanisms in general has to be given first. There are mainly two approaches in existence. The first one is using a traditional byzantine fault tolerant algorithm like *Practical Byzantine Fault Tolerance* (PBFT) [16]. PBFT employs a voting system in order to decide on a block and is resilient against a maximum of 33% of the participating nodes being malicious. However, this system needs some central authority to manage network membership and the voting process does not scale well with large network sizes. Nevertheless PBFT is still a decent choice and for example was adopted for Hyperledger Fabric [7]. A new take on PBFT is the *Federated Byzantine Agreement* (FBA) [23]. FBA is fully decentralized and scales better. The central idea is that nodes do not rely on the majority of other nodes on the network for block acceptance any more, but on a majority of a set of nodes they trust. This mechanism is the one used by the financial service Stellar [14].

The other approach is a *leader election protocol* and also known as "Nakamoto consensus", named after the Bitcoin

creator [15]. The idea of this protocol is to somehow randomly select a leader that may then create one block. Apart from Proof of Work discussed in section 2.3 the other notable, but far from being as widely used, representative of this approach is *Proof of Stake* (PoS). The general idea of PoS is that someone who owns a large part of a system is expected to naturally act in the interest of this system. Therefore a node's chance of creating the next block should be higher, the more value they control. The crypto-currency PPCoin [22] features an implementation of this mechanism. However, it does not rely on pure PoS and combines PoS with PoW, making mining easier for richer nodes. The feasibility of pure PoS is highly controversial.

All of these mechanisms are flawed in some way, especially when applied to a private blockchain. PBFT (and most other traditional byzantine fault tolerant) algorithms generally are a secure choice for small networks. However, they scale so badly (especially PBFT) that even private networks might quickly grow too large to achieve decent performance. Leader election protocols on the contrary are highly scalable, but PoW is the only one proven and extensively tested one. And while the PoW system still is immensely popular and used by a lot of big blockchain projects, it also has several severe weaknesses. The most important ones were already discussed in section 2.3 and following. When applied to a private blockchain these problems only augment.

The first problem is mining incentivization. This might be addressed by having decent transaction fees. But even if the stakeholders of a private blockchain would agree on a mining reward this would undermine one of the technology's major benefits: The costs of maintaining the blockchain would be tremendous.

The most critical problem relates to the network's size. The small amount of nodes in a private blockchain results in less combined computational power. This makes the system vulnerable to attack such as the ones presented in section 2.5, because one individual can gain over 50% of the network's computational power with relative ease. In terms of security this is catastrophic and renders the blockchain useless. By these reasons, other consensus mechanisms are needed in smaller, private networks.

4. HYPERLEDGER SAWTOOTH

Private blockchains in general are still very early early in development and next to no commercially used private blockchain systems are known at the time of writing (mid 2017). The most promising development in private blockchains so far probably is the *Hyperledger Project* [11]. This initiative was brought to life by the Linux Foundation in 2015 and is intended to form a community developing private blockchains together. The goal is to improve on existing blockchain technology in order to make it applicable for business. Prestigious members like IBM, Intel, SAP and many more, underline the huge amount of interest in this technology.

There are several different projects being developed and all of them are open source. This paper will focus on *Sawtooth* [10] (sometimes also referred to as Sawtooth Lake), because it seems to be the furthest in development and has the best specifications [6]. It is mainly written in Python and offers APIs in C++, Go, Java, Javascript, and Python. Sawtooth is able to support permissioned, as well as permissionless networks with a maximum of about 100 nodes. These networks can be deployed anywhere.

Table 4: Sawtooth Transaction Structure [8]

Name	Data Type
header	bytes
header_signature	string
payload	bytes

Table 5: Sawtooth Transaction Header Structure [8]

Name	Data Type
batcher_pubkey	string
dependencies	repeated string
family_name	string
family_version	string
inputs	repeated string
nonce	string
outputs	repeated string
payload_encoding	string
payload_512	string
signer_pubkey	string

Hyperledger Sawtooth was proposed to be included in the Hyperledger Project by Intel in April of 2016 [9] and is an active Hyperledger project at the time of writing (mid 2017). It is meant to be an enterprise solution for not only running, but also building and deploying (private) blockchains.

Nodes in Hyperledger Sawtooth can fill the roles of clients and/or validators. Clients are individuals interacting with the system by querying the state of the system or issuing transactions. Validators are nodes involved in the consensus algorithm and can be used for clients to request inclusion of transactions into the current blockchain. These validator nodes are comparable to miners in Bitcoin. However, they do use a different consensus mechanism that is presented in section 4.6;

Sawtooth's overall architecture is quite similar to Bitcoin's and will be presented in the following sections. The technical details were taken from the Hyperledger Sawtooth Documentation [8].

4.1 Transactions

Sawtooth features not only one type of transaction, but many different ones. They are grouped up into transaction families. The project comes with three already configured transaction families: `sawtooth_config`, `intkey` and `sawtooth_validator`. Developers are able to add more custom ones in order to tailor the system to every possible use case.

The `sawtooth_config` family allows for storing configuration settings on the chain. This is extremely powerful, as this enables users to configure the chain using the chain itself. For example this includes the transaction limit per block or

doi: 10.2313/NET-2017-09-1_06

Table 6: Sawtooth Batch Structure [8]

Name	Data Type
header	bytes
header_signature	string
transactions	repeated Transaction

Table 7: Sawtooth Batch Header Structure [8]

Name	Data Type
signer_pubkey	string
transaction_ids	repeated string

the desired average time between blocks. This family also is special because it deviates from the standard consensus algorithm. Instead it has two options: Only one authorized node that is able to apply changes on it's own or a voting system for multiple authorized nodes with a configurable acceptance threshold. The second transaction family included is the `intkey` family. These transactions can be used to associate integer values with names and modify these values. This allows for easy creation of bitcoin-like blockchains for testing. The remaining set of pre-installed transactions is the `sawtooth_validator` family, which is responsible for adding new validators to the network.

A transaction in Sawtooth is composed of a header and a payload. Table 4 shows the structure of a transaction and Table 5 lists the transaction header fields. The `dependencies` field is useful for validators and lists the transactions that have to precede this one. The fields `family_name`, `family_version` and `payload_encoding` are providing information about the family (and version of that family) the transaction belongs to, as well as the family specific encoding used for the payload. The `nonce` field is just a random number used to differentiate between two transactions with the same content. The last header field of interest is the `batcher_pubkey`. This field ensures that a transaction can only ever be included in a batch by someone that was allowed to do so by the creator of the transaction. Batches are discussed in section 4.2. Security is provided by the `header_signature` found in the transaction. It can be validated using the signer's public key (`signer_pubkey` in the header) and the payload is bound to the transaction header via it's hash in the `payload_512` field.

The transaction payload consists of a list of key/value-pairs that are encoded according to the `payload_encoding` field. The number and type of these values are defined by the used transaction family.

4.2 Batches

Batches are the first major deviation from the Bitcoin blockchain presented in section 2. A batch groups up one or more transactions of any type. It is to be noted that batches do not replace blocks. Batches are a new component conceptually located in between transactions and blocks. The main benefit of a batch is that it can either be accepted or declined in all of it's entirety, but under no circumstances only part of the transactions in a batch may be accepted. This for example is useful in a scenario where two configuration changes are supposed two be made, where applying only one of these changes might have a catastrophic effect on the system.

Structurally they consist of a header and a list of transactions. This is shown in Table 6 and Table 7. The `signer_pubkey` allows to check whether all transactions were rightfully included into this batch. The transactions of the batch are bound to the batch header via their `transaction_ids`. The batch is secured by the `header_signature` that can be validated using the signer's public key (`signer_pubkey` in the header), just like a transaction is.

4.3 Blocks

One block in Sawtooth groups up transactions to be applied to the ledger just like it does in Bitcoin. The most important property to mention here is that the block does not group up the transactions directly. It only groups up batches. This means that every transaction has to be batched. If a transaction's acceptance is not supposed to depend on any other transaction(s) that still means that a batch has to be created for just this single transaction.

4.4 Global State

On top of simply storing the blocks, every Sawtooth validator also possesses a *radix merkle tree* in order to additionally store the current state of the blockchain. This again is a major difference in comparison to the Bitcoin blockchain. It has the benefit of being able to query the current state without having to compute it from all of the blocks first. This radix merkle tree is separately constructed by every validator and never shared via network.

Every piece of data stored in this tree has a unique address defined by the family and the piece of data. For example, the address that defines the maximum number of transactions per block is `sawtooth.validator.max_transactions_per_block`. The actual address used by the tree is a radix address derived from this address. These radix addresses are 35 bytes long. While the first 3 bytes of the radix address are the first 3 bytes of the `SHA256` hash of the namespace address, the other 32 bytes of the radix address have to be implemented specifically for every transaction family. At each node of the tree there is one byte labeling all outgoing edges. Following the bytes of a radix address leads to the leaf node containing the corresponding piece of data. This enables very efficient lookup operations.

Additionally the tree is also a merkle tree, meaning that all nodes can be hashed up into a single root hash. Each block header contains the root hash of the according tree. This links the blocks with the tree and assures that there also is consensus on the current state.

4.5 Journal

The Journal is the central component of every validator. It is highly modular to allow for different consensus algorithms. It is responsible for generating new blocks, validating candidate blocks received via network and evaluating every valid block in order to determine whether it should be the new chain head. It also stores all blocks of the current chain without any forks.

doi: 10.2313/NET-2017-09-1_06

Whenever the Journal receives a batch it first waits for all dependencies to have been processed. Then it validates the batch and places it in storage where it waits to be included in a block. Periodically the Journal tries to build a new candidate block from the stored batches. If successful, the Journal waits until it is allowed to publish the block by the consensus algorithm (refer to section 4.6) and then proceeds to do so.

When the Journal receives a block, it is processed as soon as all predecessors are present. The following steps are taken: At first the root of the fork is determined. If the block simply extends the last block, the current chain head is the fork root. Then from this fork root on, all successors up until the new block are validated. In the end, it has to be decided whether the new block should become the new chain head. This decision is entirely dependent on the used consensus algorithm. If the new block is accepted as the new chain head, the block store is updated and all batches included in this block are removed from the pending batches.

The block validation process begins with a check for formal completeness. This includes existence of a valid predecessor as well as the presence of all batches on the block and the correct batch order. Next, all batches are validated. There may neither be duplicate batches, nor transactions and all of the transactions need to have valid dependencies. Then the block is validated according to consensus rules. Finally, the state root hash present in the block header is checked against the locally computed one.

Processing blocks inside the Journal is asynchronous. This improves performance while under great load.

4.6 Consensus using Proof of Elapsed Time

Because of all the problems PoW has, Intel decided to come up with their own leader election protocol: *Proof of Elapsed Time* (PoET) [27]. Although Sawtooth can theoretically support other consensus mechanisms, PoET is the most prominent (and currently only) one. The basic idea of this system is that every validator generates a random wait time for every new block it wants to create. The first validator who's timer expires is allowed to create the block. This way every CPU has the same chances of winning this lottery. Proof of Elapsed Time is a huge leap forward in blockchain technology, because it features all the properties of Proof of Work without the huge waste of power.

In order to guarantee the security of this system, it is necessary to be able to verify the correct execution of the timer process. This is realized by using Intel's *Software Guard Extension* (SGX) [13] [17]. Since the Skylake generation of Intel processors, this feature is becoming available on more and more Intel processors. SGX runs programs in protected areas of memory called enclaves. In this enclave the correct execution of a program can be cryptographically attestated and neither the user, nor the operating system is able to read or manipulate the enclave. Even though the initialization of an enclave is performed by untrusted software, it is possible to verify that the enclave was initialized properly. A *measurement hash* generated during initialization can prove that the enclave was set up correctly.

The backbone of this system's security is that a remote party can verify the integrity of an enclave. Every chip manufactured by Intel receives two secrets into the chip's one-time programmable registers: The *Provisioning Secret* and the *Seal Secret*. While the first one is known to Intel's *Provisioning Service*, the latter one is generated on the chip and not known by anyone. This is beneficial should Intel ever be the target of a successful hack, because the Seal Secret is used to derive a key that is used for securely storing other used keys in the system. With a *Provisioning Key*, generated from the Provisioning Secret, the enclave's certificate-based identity (created by the enclave's author) and a security version number, the *Provisioning Enclave* can request an *Attestation Key* from the Provisioning Service. After provisioning the local SGX, the *Quoting Enclave* can be used for software attestation. It receives a local attestation report from the enclave running PoET and signs it with the Attestation Key. The signed *attestation* (also called quote) is the proof for a newly created block. This whole system is completed by the *Intel Attestation Service* (IAS) [1], a web service that verifies the attestation issued by a node's enclave.

However the PoET system is still experimental and has not yet been fully implemented. Documentation is also lacking to non-existent, making it very hard to go into detail. At the time of writing this paper (mid 2017) only a test implementation exists that is not safe for use in a production environment, because it only uses a simulated enclave. All details mentioned here should be taken with a grain of salt.

4.7 Problems with Proof of Elapsed Time

PoET in the currently envisioned version requires several Intel services. This can be seen as a setback to the completely decentralized nature of a blockchain, as Intel is acting as a central authority and every node has to trust them.

Another disadvantage introduced by the IAS is an increase in difficulty of verifying blocks in comparison to PoW. While the latter one only requires computing a hash and comparing it, this new one needs to query the IAS. This adds a full round trip time to every verification.

The SGX is theoretically breakable with a realistic amount of effort, resources and knowledge. The result is catastrophic because with PoET a single broken node is able to perform majority attacks on the blockchain, such as the ones discussed in section 2.5. Intel is working on detecting compromised nodes through statistical analysis, but the attainable effectiveness remains to be seen.

Further problems and vulnerabilities may become apparent once PoET has been fully implemented and thoroughly tested. Before that it will not be possible to accurately assess the suitability of PoET as a distributed consensus mechanism.

5. RELATED WORK

Apart from Hyperledger Sawtooth, that was presented in section 4 of this paper, there are some other development efforts in the field of private blockchains worth mentioning.

Hyperledger Fabric [7] also is a part of the Hyperledger Project. Fabric was proposed for Hyperledger in early 2016

doi: 10.2313/NET-2017-09-1_06

by IBM and Digital Asset. It provides a foundation for developing permissioned blockchains using PBFT consensus.

Ripple and *Stellar* are two big financial service providers relying on their own permissioned blockchains. They connect financial institutions like banks with their network, allowing for fast and secure transactions in between them. These transactions use their respective crypto-currency to make simple currency exchange possible. The big difference in between these two is, that Stellar is a nonprofit organization, while Ripple is not. In terms of technology, Ripple uses a custom voting algorithm called *Ripple Protocol Consensus Algorithm* (RPCA) [26] and Stellar employs FBA [23].

6. CONCLUSION

This paper gave an introduction to blockchains using the example of Bitcoin. It was shown that PoW, being the most used public consensus algorithm, is effective at securing a distributed ledger if the network is large enough. Yet it is far from efficient due to the enormous power consumption introduced by mining. It was further argued that PoW simply is not applicable for small networks, such as private blockchain systems, because they extremely facilitate 50%-attacks.

Hyperledger Sawtooth by Intel was found to be a promising project to become a breakthrough in the field of private blockchains. The most important innovation is the new leader election protocol in development called Proof of Elapsed Time. It theoretically provides the same benefits as PoW does, but is applicable to smaller networks and does not nearly consume as much power. All of this comes at a price though: Because of it's reliance on Intel's trusted execution environment SGX, the algorithm requires nodes to place at least some trust in Intel. Also the block verification is slightly more complex, because it requires a query to the IAS.

Even though PoET is showing a lot of potential, it is far too soon for a final verdict. The actual viability remains to be seen once Sawtooth is production ready.

7. REFERENCES

[1] Attestation Service for Intel Software Guard Extensions (Intel SGX): API Documentation. `https://www.hyperledger.org/projects/sawtooth`, retrieved June 2017.

[2] Bitcoin Developer Reference. `https://bitcoin.org/en/developer-reference`, retrieved June 2017.

[3] Bitcoin Wiki: Block size limit controversy. `https://en.bitcoin.it/wiki/Block_size_limit_controversy`, retrieved June 2017.

[4] Coin Dance: Bitcoin Statistics. `https://coin.dance/stats#blockchain`, retrieved July 2017.

[5] Ethereum Website. `https://www.ethereum.org/`, retrieved June 2017.

[6] Hyperledger Comparison Guide. `https://github.com/hyperledger/hyperledger/blob/master/hyperledger_Overview_Comparison_Guide.md`, retrieved June 2017.

[7] Hyperledger Fabric Website. `https://www.hyperledger.org/projects/fabric`, retrieved June 2017.

[8] Hyperledger Sawtooth Documentation. `http://intelledger.github.io/`, retrieved June 2017.

[9] Hyperledger Sawtooth Proposal. `https://docs.google.com/document/d/1j7YcGLJH6LkzvWdOYFIt2kpkVlLEmILErXL6t-Ky2zU/edit`, retrieved June 2017.

[10] Hyperledger Sawtooth Website. `https://www.hyperledger.org/projects/sawtooth`, retrieved June 2017.

[11] Hyperledger Website. `https://www.hyperledger.org/`, retrieved June 2017.

[12] Hyperledger Website: About. `https://www.hyperledger.org/about`, retrieved June 2017.

[13] Intel SGX Developer Zone. `https://software.intel.com/en-us/sgx`, retrieved June 2017.

[14] Stellar Website. `https://www.stellar.org/`, retrieved June 2017.

[15] M. Ali and J. Nelson. Blockstack: A Global Naming and Storage System Secured by Blockchains, 2016.

[16] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance, 1999. `https://www.usenix.org/legacy/events/osdi99/full_papers/castro/castro_html/castro.html`, retrieved June 2017.

[17] V. Costan and S. Devadas. Intel SGX Explained, 2016.

[18] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juelsand, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. On Scaling Decentralized Blockchains, 2016.

[19] S. Deetman. Bitcoin could consume as much electricity as Denmark by 2020, March 2016. `https://web.archive.org/web/20160828092858/http://motherboard.vice.com/read/bitcoin-could-consume-as-much-electricity-as-denmark-by-2020`, retrieved June 2017.

[20] I. Eyal and E. G. Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable, 2013.

[21] G. O. Karame, E. Androulaki, and S. Capkun. Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin, 2012.

[22] S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012.

[23] D. MaziÃĺres. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, 2015.

[24] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[25] M. Rosenfeld. Analysis of hashrate-based double-spending, 2012.

[26] D. Schwartz, N. Youngs, and A. Britto. The Ripple Protocol Consensus Algorithm, 2014.

[27] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. van Renesse. REM: Resource-Efficient Mining for Blockchains, 2017.

Challenges for Modelling of Software-based Packet Processing in Commodity-Hardware using Queueing Theory

Christian Thieme
Advisor: Daniel Raumer
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services SS2017
Departments of Informatics, Technical University of Munich
Email: christian.thieme@tum.de

ABSTRACT

In the past few years, a trend of softwarization in the networking world has emerged. Software defined networking and the virtualization of networking functions, which can be realized in off-the-shelf hardware, help network operators to meet the new challenges that arise from nowadays traffic demands. The demand on routing traffic can partly be traced back to the rise of a digital society and hypes like internet of things, streaming platforms and others.
In order to encounter these challenges software defined networking and virtualized networking functions are required to be implemented efficiently on commodity hardware, so they can offer a high quality of service, while also saving expenses for hardware and energy consumption. These and other reasons motivate modelling the system of commodity hardware devices.
This paper introduces the basics of Queueing Theory Models and then identify challenges which arise from modelling commodity hardware. Furthermore, recent literature is surveyed in respect to how or whether the proposed models map hardware and software properties.

Keywords

Queueing Theory, NFV, network function virtualization, Model, SDN, software defined networking

1. INTRODUCTION

The internet has led to the creation of a digital society, where almost everything is connected and accessibly from anywhere [11]. This effect has also been intensified by the rise of the internet of things. The demand on processing traffic has therefore risen significantly. To meet those demands while saving costs (e.g. costs for hardware, software and energy consumption) and staying environmentally aware, new approaches to meet these demands have been developed.

One step to engage these challenges is software-defined networking (SDN). With this new paradigm, the control plane and the data plane are decoupled and are more bundled within single networking devices. In SDN the control plane decides how to handle network traffic. While the data plane, which forwards traffic according to decisions made by the control plane, can be leveraged and managed in a simpler way (i.e. network switches become simple forwarding devices). [11]

At the same time, a complementary trend has emerged: network function virtualization (NFV). In NFV networking functions like packet-forwarding, routing, applying firewall rules etc. are decoupled from their physical devices. Furthermore, NFV has the ability to facilitate the new deployment of new services with increased agility and faster time-to-value [15].

Both paradigms, namely SDN and VNF, offer the opportunity to be flexible and considerably cheap compared to dedicated hardware. Both approaches allow well-designed networking-systems to meet and balance demands, while staying cost-efficient and maintaining a high quality of service (e.g. for an ISP). In order to develop, optimize and understand certain behaviours of the new systems, modelling is crucial. "Modeling is an important and useful approach for performance evaluation and system validation and it can provide prediction and comparison of design alternatives" [1].

The focus of this paper lies on queueing theory models and is structured as follows: an introduction to the basics of queueing theory is presented in chapter 2. Chapter 2 also provides insight on how a device (i.e. a router) can be mapped to a queueing theory model. Furthermore, Chapter 2 provides reasons and properties, why queueing theory can be an appropriate tool to model networking systems or devices. Chapter 3 lists some challenges and real-world effects that might come with modelling commodity hardware computers. Chapter 4 covers recent literature, in which queueing models have been developed and surveys them regarding the collected challenges from chapter 3. Chapter 5 concludes this paper and summarizes the results of this work.

2. QUEUEING THEORY

"Queueing theory deals with one of the most unpleasant experiences of life, waiting. Queueing is quite common in many fields, for example, in telephone exchange, in a supermarket, at a petrol station, at computer systems, etc." [18]

It was mentioned first by A.K. Erlang at the beginning of the 20th century, when he studied the behaviour of telephone networks [5, 6]. Erlangs works inspired engineers and mathematicians to deal with queueing problems using probabilistic methods [18]. And now, over a hundred years

doi: 10.2313/NET-2017-09-1_07

later, hundreds of articles and books have been published and queueing theory is still a widely-used modelling technique.

Queueing theory is used to compute mean values and predictions of a system. This chapter is mostly inspired by the books of Kobayashi et al [10] and Sztrik [18].

2.1 Basic Queueing Theory

To characterize a queueing system, the probabilistic properties of the incoming flow of requests, service times and service disciplines have to be identified [18]. The time between incoming requests is also called interarrival time. In the following the term customer is used to describe an entity that is being served or processed by a service or server [10]. The interarrival time A(t) is usually given as a stochastic distribution, which can, in example, express the probability of customers arriving to our system within a specific time t.

$$A(t) = P(\text{Customer enters system} < t)$$

There is also the property of service time. The service time B(x) is a probabilistic distribution, and denotes, for how long a request is served. Following the example from above, this distribution expresses the probability of a customer being served within time x.

$$B(x) = P(\text{Customer is served} < x)$$

The interarrival times and service times are commonly supposed to be independent random variables [18]. The service discipline describes in which manner the requests are being processed. Some commonly used service strategies are:

FIFO: First In First Out
The first to arrive at the queue, is served first.

LIFO: Last In First Out
The last to enter the queue, is served first.

PS: Processor Sharing
Entities arriving at the Service are served immediately and receive service in the following way: C/n. Where C equals the total servicing capacity of the server and n is the number of entities. This is an idealization of the Round-Robin scheduling scheme. [10]

Priority: Some other priority, i.e. Random
Enqueued entities are being processed according to some other priority.

Queueing Systems can have multiple service units offering the same service that serve customers. The number of service units within a queueing system is denoted as m.

Finally, the capacity of the system and the population size of the queueing system can be identified. The capacity K of the system represents the maximum number of customers within the system, including the ones being served [18]. The

Figure 1: A typical depiction of a single queue, with arrival rate λ and a single service node with service rate μ.

population size n basically describes whether the source of incoming customers is limited by a variable, number or ∞.

Figure 1 shows the usual visual representation of a simple single queueing system. Customer arrive at arrival rate λ and are enqueued into the waiting area. The customers are served at rate μ.

2.2 Kendall's Notation

In 1953 Kendall [9] introduced a notation to describe a queueing system, which is commonly used nowadays. He denoted the system by

$$A \;/\; B \;/\; m \;/\; K \;/\; n \;/\; D$$

where

A: distrubtion function of the interarrival times,

B: distribution function of the service times,

m: number of service units (offering the same service),

K: capacity of the system,

n: size of the source, and

D: service discipline.

Throughout literature, the notation A/B/m is used when the capacity of the system and the size of the source are assumed to be ∞ and the service discipline is FIFO. Commonly used options for A and B are:

M: A Poisson arrival distribution (an exponential distribution) or an exponential service time distribution. This usually denotes a Markovian process, which possesses the memoryless property. The memoryless property means, regardless of how long ago a customer arrived, the probability of a customer arriving now stays the same. [10]

D: A deterministic or constant value.

G: A general distribution with a known mean and variance formulas. [10]

Example 1: M/M/1/∞/∞/FIFO is abbreviated to M/M/1, denoting a system with poison arrivals, exponentially distributed service times and a single service unit.

doi: 10.2313/NET-2017-09-1_07

Example 2: M/G/m denotes an m-server system with Poisson arrivals and generally distributed service times.
Example 3: M^x/G/M denotes the same system as in Example 2, but additionally defines that there are x sources from which customers can arrive with the same arrival distribution.
Example 4: M/G/m-S denotes the same system as in Example 2, but additionally defines the queue length S.
Example 5: M/M/r/K/n describes a system with a finite-source of n elements, where they stay for an exponentially distributed time and the service times are exponentially distributed. Furthermore the service is carried out according to the request?s arrival by r servers, and the system capacity is K. [18]

More examples, explanations and theory can be found in [18] and [10] or any other book on queueing theory.

2.3 Properties

From the queueing system the following properties can be derived:

- The arrival rate of customers (denoted by λ) which can also be referred to as birth rate.
- The service rate (denoted by ţ) which can also be referred to as death rate.
- The server utilization, throughput and response time.
- The number of customer in the system at time t.
- The probability of n costumers in the system at time t.
- The length of an idle or a busy period.
- The traffic intensity (usually denoted as $\rho = \lambda \ / \ \mu$), which represents the expected number of arrivals during the service of a customer.

Furthermore, there are some more properties that can be calculated only if the the system is in steady-state. This is the case, if the system is not overloaded, e.g. when requests arrive faster than the system can processes them. The steady-state is also called equilibrium. "A queue is defined as stable if $\mu > \lambda$ or $\rho > 1$. When $\rho > 1$, requests entering the queue accumulate faster than they can be served and, in theory, latency increases to infinity. In practice, the system saturates, limiting the request rate to be $\lambda = \mu$, (or $\rho = 1$) and stabilizes the queue with high latency." [16]

Using theorems and formulas, the following properties of a M/M/1, M/M/n or M/M/∞ queueing system can be derived if the system is in steady state:

- The mean number of customers in the queue.
- The mean number of customers in the system.
- The mean total waiting time in the system.

3. CHALLENGES IN MODELLING X86 DEVICES USING QUEUEING THEORY

This chapter lists and discusses some features that can come with commodity hardware. Software effects are also considered. Each of the features is explained and if necessary an example is given. Then an it is pointed out, how the features could be taken into account by a queueing theory model. At the end, there is also an example given, how a commodity hardware computer could be modelled as a queueing system.

3.1 Multi-Core

One of the main reasons, why off-the-shelf hardware is able to compete with dedicated hardware is because of the ability to process data in parallel due to multiple central processing units (CPUs). Cores or threads are mapped to service units in every queueing model and therefore represented by the number of service units. The CPU is therefore modelled as a queueing system.

Furthermore the CPU-frequency is modelled and represented by the service time distribution. The frequency usually has the biggest impact on the service time distribution.

3.2 Bus

Data from any NIC usually has to be transferred to memory or cache, so it can be accessed and processed by a processing unit. Direct Memory Access (DMA) is a way to transfer data from a device via a bus system to the main memory without involving the CPU. This provides an efficient way of moving data into the main memory for processing. [17]

A step further than DMA is the direct cache access (DCA), which basically does the same as DMA, just that the data is directly passed to a CPU cache. Though, these techniques are not available on every off-the-shelf hardware. In any case, a bus system moves the data into the cache or main memory.

The bus system, which transfers the data to the cache or main memory can be modelled as a single queue. Therefore, the queueing system of the bus and the queueing system of the CPU can be modelled as a queueing network [1].

3.3 Number of NICs

Using commodity hardware, systems have to be equipped with Network Interface Cards (NICs) in order to receive and send packets. To fully utilize the system, as many NICs as supported should be installed. As shown by Runge et al. in [19], the number of NICs can have a significant impact on the performance of a networking device.

As NICs are part of a computer, their output is an input to the next queueing system within the computer (e.g. a bus or CPU). A NIC itself can be modelled as a single queueing system. The input from NICs can be implicitly modelled by adding an exponent to the interarrival distribution of the next queueing system entity in the queueing network (e.g. M^x/M/1).

[1]When multiple queues are plugged together, e.g. the output of a queue is the input of the next queue, then this is called a queueing network.

doi: 10.2313/NET-2017-09-1_07

3.4 Memory and Software Latencies

There are several latency effects that can arise from memory and software. In the following some effects which can introduce latencies are listed:

- moving data between cache hierarchies (L1, L2, L3 and main memory)
- interrupts, which can be triggered by hardware (e.g. a packet arrives at a NIC and need to be stored in main memory) or software (e.g. a process with higher priority forces a context switch)
- resource contention (e.g. threads are fighting over CPU time and force context switches)
- queue implementation (e.g. a queue must be implemented in a way so that parallel processing, retrieving and inserting of data is efficient, which is also pointed out by Orozco et al. in [16])

These latencies have an impact on the time a packet is spending in the system. One could model these effects by adding a term or constant value to the service time distribution. Even though, the listed effects are usually neglected for simplicity by the models proposed in literature.

3.5 Finite Memory

Any physical device has limited resources regarding space (caches, buffers and main memory) available. And since they represent queue sizes (and capacity of the system) they should be modelled and not assumed to be limitless. A infinite or infinite buffer has a direct impact on properties like the waiting time for a packet in the system. Though, most proposed models neglect this fact. The capacities of the system are assumed to be infinite.
When queueing models in networking assume infinite buffers, they can become vulnerable to attacks (i.e. Denial of Service Attacks) as shown by [3].

3.6 Batch-Processing

In computer hardware, it is common to load and transfer batches of data in order to hide loading latencies. I.e. a NIC does not instantly transfers a packet on its arrival, but waits until a certain time has passed or the batch is filled up. Also CPUs tend to load blocks of data into higher cache hierarchies at once for processing.This behavior can influence the performance of a system significantly. Consequently, batch processing has an impact on the time a packet stays within the system and the service time of a packet. Batch processing is sometimes modelled by recent literature.

3.7 Packet Size

When dealing with traffic in the internet, it is very likely to encounter packets of different sizes. I.e. when processing a big packet (or a batch of big packets), the total number of packets processed is comparatively lower, compared to processing small packets. On the other hand, the throughput (in Mbit) is higher that way.

Therefore, the size of different packet sizes should be considered in the service time of the queueing model. This also has an impact on the time the packet spends in the system. The packet size is usually not considered in recent models.

3.8 Example

In general, commodity hardware can be modelled as a tandem queueing network [14]. A tandem queueing network is a subclass of queueing networks. In case of a tandem queueing network, customers, who arrive at the system, can encounter multiple queue-service pairs, until they leave the system again. E.g. this means that after a packet has entered a queue and has been serviced, it enters yet another queue to a service and so on, until it finally leaves the system again. It should also be noted, that customers cannot skip one or multiple queues.

In literature, a computer system is modelled either as a single queue, usually representing a bottleneck, or as a network of queues. In Figure 2, an example of computer modelled as a queueing network is shown. The figure shows the packet flow $\lambda+$ through the computer. λ represents the incoming

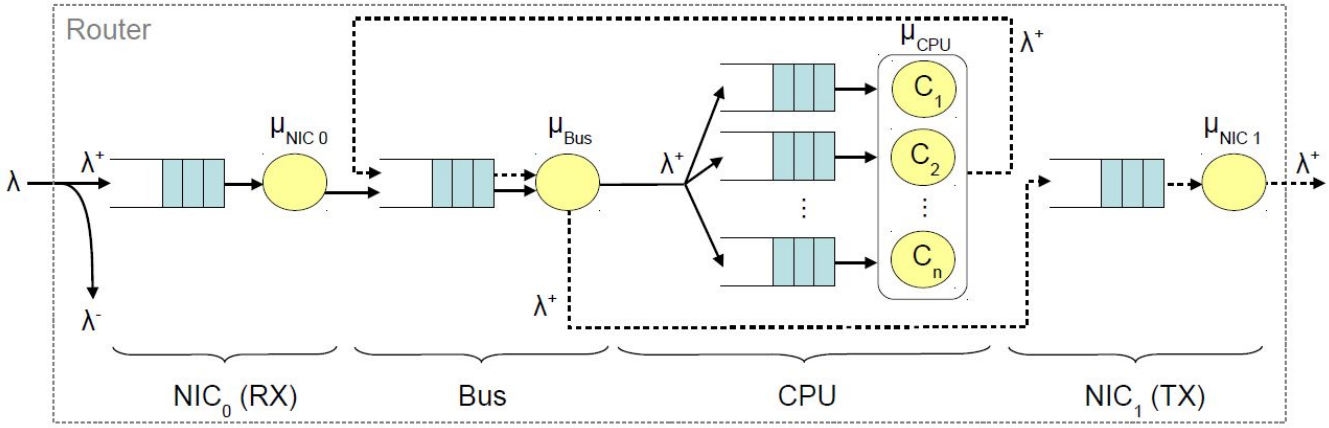


Figure 2: A Router Model of [14] which shows, the flow of packets through a computer. The NICs, bus and CPU are each modelled as a queueing system, forming a queueing network.

doi: 10.2313/NET-2017-09-1_07

traffic which is split into λ+ (packets that enter the system) and λ- (packets that are dropped at the NIC due to overload). Solid arrow depict the packet flow towards the CPU, while dashed arrows depict the flow of processed (outgoing) traffic. Mayer et al [14] show this figure, but finally model only the CPU as an M/M/n queue.

A possible alternative for the modelling of Figure 2 could be as follows: The system can be modelled as a queueing network, starting with an M/M/1 queue, which represents the NIC that receives the incoming traffic. Packets leaving the NIC are then enqueued into another M/M/1 queue for the bus system. The CPU can be modelled either as n * M/M/1 queues or as M/M/n queue, where n is the number of CPU cores or threads. Finally, packets leaving the system must pass through the outgoing NIC, which again could be modelled as an M/M/1 queue. The here proposed queueing network results in a jackson network [2].

4. QUEUEING THEORY MODELS IN X86 INTERCONNECT DEVICES

In this chapter, some recent literature is presented and surveyed in respect to whether the proposed models consider the aspects presented in chapter 3, or not. The focus is on research, which uses commodity hardware. At the end of this chapter a comprehensive overview within Table 1.

Meyer et al. [14] propose a tandem queuing model to model a software router. This model is then simplified to the extend where the model basically only includes the packet size and multiple service units (CPU cores) and the services times. This is done in order to measure the performance at the bottleneck, namely the CPU. Their goal is to measure and predict the maximum throughput of a general software router.
The authors do not provide a notation for their queueing system, but given from the information in the text, it boils down to a M/M/n queue for the entire router. In the formulas, the packet size and multi-cores are taken into account. Other effects are neglected or ignored for simplicity.

In the work of Jemaa et al. [2] a model for VNF placement and provisioning optimization strategies over an edge-central carrier cloud infrastructure is introduced, which takes QoS requirements into account. Since the paper is dedicated to spawning VNFs in cloud environments, it could be considered off-topic. Nonetheless, the reason this paper was taken in, was because the paper includes the delay, which introduced by virtualization layer of a virtual machine (VM) in the cloud.
The authors model each VNF as a single M/M/1 queue. The VNFs are hosted in a VM, which is therefore modelled as a jackson network. Packets that are transferred between two VMs are modelled as a M/GI/$infty$ queue which acts like a bus. Arriving packets at the queue are instantly processed and sent to the next VM. In the formulas the latencies of spawning a VNF is taken into account.

An energy-aware resource allocation scheme to manage virtual machines, dedicated to perform certain virtualized network functions is proposed by Bruschi et al. in [4]. The authors M^x/M/1/SET model assumes batch arrivals with a constant packet size and take this into account in their formulas. They also model multiple packet arrival sources and a wake up time for servers that have been idle.

Gebert et al. [7] propose a general analytical model for generic virtualized network functions running on commodity hardware. They model NICs as a external queues which send incoming traffic into one GI^x/GI/1-L queue, which represents the CPU, using batch-processing. A packet in the batch arriving at the CPU queue is dropped if the packet has stayed in queue for too long. Packets are also rejected, when they arrive and would have to wait longer than L-1 packets need for processing.

Jarschel et al. [12] derive a basic model for the forwarding speed and blocking probability of an OpenFlow switch combined with an OpenFlow controller. In their model, they assume a queue of infinite size, which collects incoming traffic. That traffic is then forwarded and passed into a queue with limited buffer size (denoted by the parameter S in their queueing system).

In [8] Mahmood et al. extends the case in which a controller in a OpenFlow network is only responsible for a single node in the data plane to multiple nodes. They approximate the case as an open Jackson network and provide formal proof for their approximation for the case of infinite buffer and finite buffer sizes.

[2] A jackson network is a special case of queueing networks where all external arrivals into the queueing system are a Poisson Process, service times are exponentially distributed, and the utilization of all queues is less than one [18]

Table 1: Summary of the surveyed papers in respect to whether they take the challenges of chapter 3 into account, or not.

	3.1 Multi-Core	3.2 Bus	3.3 Number of NICs	3.4 Memory & Software Latencies	3.5 Finite Memory	3.6 Batch Processing	3.7 Packet Size	Queueing System(s)
[14]	yes	no	no	no	no	no	yes	M/M/n
[2]	yes & no	yes	no	yes	no	no	no	N * M/M/1, M/GI/∞
[4]	no	no	no	yes	no	yes	yes	M^x/G/1/SET
[7]	no	no	yes	no	no	yes	yes	GI^x/GI/1-L
[12]	no	no	no	no	yes & no	no	no	M/M/1, M/M/1-S
[8]	no	no	no	no	yes & no	no	no	M/M/1

doi: 10.2313/NET-2017-09-1_07

As can be seen in table1 the models in recent literature do not take all challenges listed in Chapter 3 into account. In most papers the packet-processing system was modelled as a single queue. This might be an sufficient assumption as long as the CPU stays the bottleneck of such systems.

5. CONCLUSION

This paper introduced basic queueing theory and provided challenges on how such a model can be applied to a commodity hardware computer. Queueing theory has proved to be a suitable model to predict long-term and average behaviours.

Judging from the validation of the proposed models that have been conducted in the presented papers, the models seem to be approximating the real world systems sufficiently. Which is impressive, given the fact that most of the models only take some of the challenges listed in Chapter 3 into account. This on the other hand, just shows how powerful this modelling technique is.

It general, is also reassuring to see that research can already provide queueing theory models for recent advances in the softwarization hype, e.g. towards flexible and scalable networking using SDN and VNF.

6. REFERENCES

[1] S. Balsamo and A. Marin. Performance engineering with product-form models. In S. Kounev, V. Cortellessa, R. Mirandola, and D. Lilja, editors, *ICPE'11*, page 437, [New York], 2011. ACM.

[2] F. Ben Jemaa, G. Pujolle, and M. Pariente. Analytical models for qos-driven vnf placement and provisioning in wireless carrier cloud. In M. . C. COMMITTEE, editor, *MSWIM 16 19TH INTERNATIONAL CONFERENCE ON MODELING, ANALYSIS AND SIMULATION OF WIRELESS AND... MOBILE SYSTEMS*, pages 148–155, [S.l.], 2016. ACM.

[3] D. S. Berger, M. Karsten, and J. Schmitt. On the relevance of adversarial queueing theory in practice. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):343–354, 2014.

[4] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo. Joint power scaling of processing resources and consolidation of virtual network functions. In *Proceedings, 2016 5th IEEE International Conference on Cloud Networking*, pages 70–75, Los Alamitos, California, 2016. IEEE Computer Society, Conference Publishing Services.

[5] A. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B 20*, B 20:33–39, 1909.

[6] A. Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *The Post Office Electrical Engineers' Journal*, 10:189–197, 1918.

[7] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia. Performance modeling of softwarized network functions using discrete-time analysis. In *2016 28th International Teletraffic Congress (ITC 28)*, pages 234–242. IEEE, 2016.

[8] M. Jarschel, O. Østerbø, A. Chilwan, and K. Mahmood. Modelling of openflow-based software-defined networks: The multiple node case. *IET Networks*, 4(5):278–284, 2015.

[9] D. G. Kendall. Some problems in the theory of queues. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(2):151–185, 1951.

[10] H. Kobayashi, B. L. Mark, H. M. Kobayashi, and analysis. *System modeling and analysis: Foundations of system performance evaluation / Hisashi Kobayashi, Brian L. Mark*. Pearson Education Ltd, London, pearson international ed. edition, 2009.

[11] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[12] M. Jarschel and S. Oechsner and D. Schlosser and R. Pries and S. Goll and P. Tran-Gia, editor. *2011 23rd International Teletraffic Congress (ITC): Modeling and performance evaluation of an OpenFlow architecture*, 2011.

[13] T. M. Runge, B. E. Wolfinger, S. Heckm?uller, and A. Abdollahpouri. A modeling approach for resource management in resource-constrained nodes. *Journal of Networks*, 10(01), 2015.

[14] T. Meyer, F. Wohlfart, D. Raumer, B. E. Wolfinger, and G. Carle. Validated model-based performance prediction of multi-core software routers. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 37(2), 2014.

[15] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. de Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[16] D. Orozco, E. Garcia, R. Khan, K. Livingston, and G. R. Gao. Toward high-throughput algorithms on many-core architectures. *ACM Transactions on Architecture and Code Optimization*, 8(4):1–21, 2012.

[17] A. Rubini and J. Corbet. *Linux device drivers: Chapter 13: mmap and DMA - Direct Memory Access and Bus Mastering*. O'Reilly & Associates, Sebastopol, 2nd ed. edition, 2001.

[18] J. Sztrik. *Basic Queueing Theory: Foundations of System Performance Modeling, pages 7-15*. GlobeEdit, Saarbrücken, 1. auflage edition, 2016.

[19] Torsten M. Runge and Bernd E. Wolfinger. How do multiple network cards influence the software router performance?

Network Anomaly Detection

An Trung Tran
Advisor: Marcel von Maltitz, Stefan Liebald
Seminar Innovative Internet Technologies and Mobile Communications SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: antrung.tran@tum.de

ABSTRACT
In this paper we present the foundations of Network Anomaly Detection, which include the definition of a Network Anomaly Detection System, its purpose, challenge. This paper also provides an overview over the different types of attributes extractable from raw network data. Another valuable aspect of this paper is the taxonomy of various different algorithm types, which are described, in a concise way. This includes the main advantages and drawbacks of each type and an evaluation of the methods as well as two examples of algorithms.

Keywords
Network Anomaly Detection, Feature Selection, Algorithms, Network Intrusion Detection

1. INTRODUCTION
According to Anderson [2], an intrusion attempt or threat is defined as "deliberate and unauthorized attempt to (i) access information, (ii) manipulate information, or (iii) render a system unreliable or unusable". With the steady advance of network-based computer systems and modern technologies, there is a increasing need of systems capable of detecting network intrusions, which pose a massive security risk. An example would be the massive Distributed Denial of Service (DDoS) attack on the french web hoster OVH in 2016. This was done by an Internet of Things-based botnet [5], which is a network of private computers infected with malicious software and controlled as a group without the owners' knowledge.

In this paper, we explore one of the measures used to counter such issues, namely Network Anomaly Detection (NAD). NAD is defined as identification of events, which differ from an expected pattern, particularly in network data. Henceforth, in the subsequent parts of this paper, we discuss the different features of the NAD as well as the algorithms and methods applied on the system.

2. NETWORK ANOMALY DETECTION
[4] defines anomaly detection in networks as "the problem of finding exceptional patterns in network traffic that do not conform to the expected normal behaviour". The outstanding patterns are mostly referred to as 'anomalies' or 'outliers' in this context. There are three types of anomalies: (i) point, which is an instance of individual data that has been found anomalous, e.g. a purchase with large transaction value; (ii) contextual, a data instance found anomalous in a specific context, e.g. a large spike in traffic in the middle of the night and (iii) collective, a collection of data instances found anomalous, e.g. breaking rhythm in an electrocardiogram (ECG) [4]. Anomalies can be divided into two categories [14]: (a) performance related anomalies and (b) security related anomalies. In this paper, we focus on the security related anomalies which occur due to malicious activities intended to flood the network with unnecessary traffic hijacking the bandwidth and turning the system inaccessible, i.e. a DDoS attack as mentioned before.

2.1 Properties of NADS
Network Anomaly Detection Systems (NADS) serve the main purpose of processing network data by monitoring packets on the network and look for patterns and is used to determine whether the input data is an anomaly or a normal data instance. According to [4], NADS is based on five different characteristics which describe the concept: (i) "Principal assumption: All intrusive activities are necessarily anomalous." (ii) The method compares the normal system state with an established profile. When the degree of deviation is too high, intrusion attempts are reported. (iii) False positives are anomalous activities, which are not intrusive. (iv) "One should select threshold levels so that neither of the above two problems is unreasonably magnified nor the selection of the features to monitor is optimized." (v) "Computationally expensive because of overhead and possibly updating several system profile matrices". Aside from that, NADS operate on three different modes: (i) supervised, which uses both training data from normal and anomaly classes; (ii) semi-supervised, which only use labeled instances of data for the normal classes and (iii) unsupervised, which requires no labeled instances of data but labeling is done by the system itself [4].

2.2 Challenge of NADS
The biggest challenge for Network Anomaly Detection Systems is the definition of the "concept of normality" [4], which is defined "by a formal model that expresses relations among the fundamental variables involved in the system dynamics" [4].

Therefore, instances are detected as anomalous, if the degree of deviation with respect to the normal profile is too high. [4] introduces an abstract model of an anomaly detection system S that uses a supervised approach. Training datasets are used and labelled for normal as well as for the anomaly class. S is defined as $S = (M,D)$ with M being the

doi: 10.2313/NET-2017-09-1_08

normal model and D being "a proximity measure that allows one to compute, given an activity record, the degree of deviation that such activities have with regard to the model M. Each system can be broken down into 2 modules: (i) a modeling module, which trains the system to achieve the normality model D and (ii) a detection module, which uses (i) to classify new traffic as anomalous or outliers. M needs to be flexible in order to handle changing scenarios".

3. FEATURES

Features are defined as attributes extractable from raw network data, in which its selection is crucial for network anomaly detection. Feature selection is the process of extracting specific features out of raw data to be loaded into an algorithm because not all algorithms work with all kinds of data. Most data can be classified into one of two groups : (a) numerical or (b) categorical. Numerical data can be divided into two more subgroups, which are discrete data (representing items which can be counted) and continuous data (representing measurements or values that can only be described using real number intervals). Feature selection offers a lot of advantages because it (i) improves the performance of an algorithm as it cuts down on feature dimensionality , (ii) removes insignificant features , (iii) improves data quality and therefore the efficiency of learning algorithms, (iv) raises the detection rate and (v) helps in understanding the data generation process as well as visualizing it [3] [4].

When looking for useful features, there are various ways to view a connection. First, it can be done by inspecting individual packets and their characteristics stored in the headers. Another method of viewing a connection could be done by analysing the packet flows from the source to destination and vice versa. In the following we are analyzing the TUIDS dataset described in [3], which divides features into three different categories: (i) packet traffic feature dataset, (ii) network flow traffic feature dataset, (iii) portscan. We are only going to analyze (i) and (ii) as the survey does not mention a significant reason to additionally consider (iii). The features listed in this dataset will be compared to other surveys the features and complemented by using other sources.

3.1 Packet Traffic Features

This section describes the possible features which can be extracted by inspecting individual packets and their headers. These are otherwise known as packet traffic features. This information can be used to create statistics to detect anomalies. Other algorithms which utilize this information to detect anomalies are available aswell. The information itself can be extracted by looking at the connection and sampling packets in different timeframes or at different points in time. Sampling is necessary here as perusal of the full packet data would be very time consuming and costly. Packet traffic features can be divided into (i) basic packet features (see Figure 1), (ii) content-based packet features (see Figure 2), (iii) time-based packet features (see Figure 3) and (iv) connection-based packet features (see Figure 4). The Figures 1-4 are giving a general overview of the features which are applicable for later uses. Besides the listed features from [3], there are other features, which can be used, when inspecting into individual packets and their headers. In comparison [11] uses IP packet size as well as TCP Header Size, TCP Window Size and TCP options and some of the

Sl.	Feature Name	Type*	Feature Description
1.	Duration	C	Time since occurrence of first frame
2.	Protocol	D	Protocol of layer 3: IP, TCP, UDP
3.	Src IP	C	Source IP address
4.	Dst IP	C	Destination IP address
5.	Src port	C	Source port of machine
6.	Dst port	C	Destination port of machine
7.	Service	D	Network service on the destination, e.g., http, telnet, etc.
8.	num-bytes-src-dst	C	Number of data bytes flowing from src to dst
9.	num-bytes-dst-src	C	Number of data bytes flowing from dst to src
10.	Fr-no.	C	Frame number
11.	Fr-length	C	Length of the frame
12.	Cap-length	C	Captured frame length
13.	Head-len	C	Header length of the packet
14.	Frag-offset	D	Fragment offset value
15.	TTL	C	Time to live
16.	Seq-no.	C	Sequence number
17.	CWR	D	Congestion Window Record
18.	ECN	D	Explicit Congestion Notification
19.	URG	D	Urgent TCP flag
20.	ACK	D	Ack flag
21.	PSH	D	Push TCP flag
22.	RST	D	Reset RST flag
23.	SYN	D	Syn TCP flag
24.	FIN	D	Fin TCP flag
25.	Land	D	1 if connection is from/to the same host/port; 0 otherwise

Note: *C-Continuous, D-Discrete

Figure 1: Basic Packet Features from [3]

Sl.	Feature Name	Type*	Feature Description
1.	Mss-src-dst-requested	C	Maximum segment size from src to dst requested
2.	Mss-dst-src-requested	C	Maximum segment size from dst to src requested
3.	Ttt-len-src-dst	C	Time to live length from src to dst
4.	Ttt-len-dst-src	C	Time to live length from dst to src
5.	Conn-status	C	Status of the connection (1-complete, 0-reset)

Note: *C-Continuous, D-Discrete

Figure 2: Content-based Packet Features from [3]

listed features. [9] used the S0 Flag, which is the first SYN packet sent, when the 3-Way-Handshake for TCP is established, as well as the rejection (REJ) flag as features and some of the listed features. In [12] identification number is used together, with acknowledgement number and the options.

3.2 Network Flow Traffic Features

This section describes the possible features which can be extracted by inspecting the flows between the source and its destination. This aspect is important as it takes a look at a collection of packets, which would allow the identification of patterns or features otherwise unnoticeable on the level of individual packets. This enables the view on stateful connections such as TCP and the TCP 3-Way-Handshake [9]. Network flow traffic features (Figure 5) are divided into (i) basic features, (ii) time-window features and (iii) connection-based features. In comparison [8] additionally counts the number of packets, acknowledgement packets, retransmitted packets and pushed packets.

Sl.	Feature Name	Type*	Feature Description
1.	count-fr-dst	C	Number of frames received by unique dst in the last T sec from the same src
2.	count-fr-src	C	Number of frames received by unique src in the last T sec to the same dst
3.	count-serv-src	C	Number of frames from the src to the same dst port in the last T secs
4.	count-serv-dst	C	Number of frames from dst to the same src port in the last T secs
5.	num-pushed-src-dst	C	Number of pushed packets flowing from src to dst
6.	num-pushed-dst-src	C	Number of pushed packets flowing from dst to src
7.	num-SYN-FIN-src-dst	C	Number of SYN/FIN packets flowing from src to dst
8.	num-SYN-FIN-dst-src	C	Number of SYN/FIN packets flowing from dst to src
9.	num-FIN-src-dst	C	Number of FIN packets flowing from src to dst
10.	num-FIN-dst-src	C	Number of FIN packets flowing from dst to src

Note: *C-Continuous, D-Discrete

Figure 3: Time-based Packet Features from [3]

Sl.	Feature Name	Type*	Feature Description
1.	count-dst-conn	C	Number of frames to unique dst in the last N packets from the same src
2.	count-src-conn	C	Number of frames from unique src in the last N packets to the same dst
3.	count-serv-src-conn	C	Number of frames from the src to the same dst port in the last N packets
4.	count-serv-dst-conn	C	Number of frames from the dst to the same src port in the last N packets
5.	num-packets-src-dst	C	Number of packets flowing from src to dst
6.	num-packets-dst-src	C	Number of packets flowing from dst to src
7.	num-acks-src-dst	C	Number of ack packets flowing from src to dst
8.	num-acks-dst-src	C	Number of ack packets flowing from dst to src
9.	num-retransmit-src-dst	C	Number of retransmitted packets flowing from src to dst
10.	num-retransmit-dst-src	C	Number of retransmitted packets flowing from dst to src

Note: *C-Continuous, D-Discrete

Figure 4: Connection-based Packet Features from [3]

Sl.	Feature Name	Type*	Feature Description
	Basic features		
1.	Duration	C	Length of the flow (in sec)
2.	Protocol-type	D	Type of protocols– TCP, UDP, ICMP
3.	src IP	C	Src node IP address
4.	dst IP	C	Destination IP address
5.	src port	C	Source port
6.	dst port	C	Destination port
7.	ToS	D	Type of service
8.	URG	D	Urgent flag of TCP header
9.	ACK	D	Ack flag
10.	PSH	D	Push flag
11.	RST	D	Reset flag
12.	SYN	D	SYN flag
13.	FIN	D	FIN flag
14.	Source byte	C	Number of data bytes transferred from src IP addrs to dst IP addrs
15.	dst byte	C	Number of data bytes transferred from dst IP addrs to src IP addrs
16.	Land	D	1 if connection is from/to the same host/port; 0 otherwise
	Time-window features		
17.	count-dst	C	Number of flows to unique dst IP addr inside the network in the last T secs from the same src
18.	count-src	C	Number of flows from unique src IP addr inside the network in the last T secs to the same dst
19.	count-serv-src	C	Number of flows from the src IP to the same dst port in the last T secs
20.	count-serv-dst	C	Number of flows to the dst IP using the same src port in the last T secs
	Connection-based features		
21.	count-dst-conn	C	Number of flows to unique dst IP addrs in the last N flows from the same src
22.	count-src-conn	C	Number of flows from unique src IP addrs in the last N flows to the same dst
23.	count-serv-src-conn	C	Number of flows from the src IP addrs to the same dst port in the last N flows
24.	count-serv-dst-conn	C	Number of flows to the dst IP addrs to the same src port in the last N flows

Note: *C-Continuous, D-Discrete

Figure 5: Network Flow Traffic Features from [3]

4. ALGORITHMS

This section presents various algorithms for NADS, categorizes them for an overview and shows two algorithms as examples. Algorithms in this specific context are mandatory as NAD is not possible without them. There are different approaches by various algorithms to solve the problem of NADS. In the end, the choice of the algorithm will influence the type and quality of the result significantly.

4.1 Classification of NAD Methods

In this subsection the different methods of Network Anomaly Detection are classified and represented as a taxonomy. An algorithm implementing classification is a classifier. In [4] are divided into 6 major categories: (i) statistical-based, (ii) classification-based, (iii) clustering and outlier-based, (iv) soft computing, (v) knowledge-based and (vi) combination learner. These 6 categories are going to be explained in detail in the following subsections in consideration of their advantages and drawbacks.

4.1.1 Statistical-based NAD

For statistical-based NAD, "an anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed" [4]. Thus, instances with a low probability of being generated are anomalies. The techniques are divided into (i) parametric and (ii) non-parametric. Parametric techniques "assume knowledge of the underlying distribution and estimate the parameters from the given data" [4], while non-parametric techniques does not. The advantage of these techniques are that they do not require "prior knowledge about normal activity" [6] and can provide accurate notification of malicious activities [4] [6]. The drawbacks are that they are vulnerable to be trained by attackers until "the network traffic generated during the attack is considered normal" [4] and setting values for different parameters and metrics is difficult, especially balancing between false positives and negatives [4].

4.1.2 Classification-based NAD

Classification-based NAD tries to assign new data instances into categories, based on training datasets [4]. Each object can be described using attributes or features. "Linear classification tries to find a line between the classes" [4], but the "classification boundary may be non-linear" too [4] as seen in Figure 6. Advantages of these techniques are that they are capable of improving their execution by integrating new data. Thus, they are adaptable for "training and testing" purposes [4]. Also these techniques have a high detection quota for known anomalies subject to suitable thresholds [4]. The drawbacks are that they are highly susceptible to the hypotheses made by classifiers. Furthermore, they are incapable of detecting unidentified anomalies until applica-

ble training datasets are provided [4].

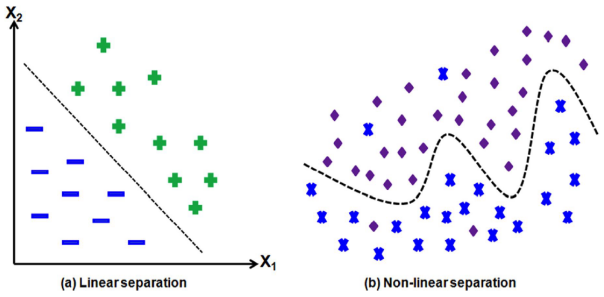


Figure 6: Classification-based NAD from [4]

4.1.3 *Clustering and outlier-based NAD*

Clustering is grouping new sets of objects into groups called clusters by using a given correlation or measuring distance [6]. Objects in the same cluster are more related to each other than those in other clusters. The most common practice "consists in selecting a representative point for each cluster" [6]. This can be visualised in Figure 7(a), which shows a set of unidentified objects in two dimensions grouped into five clusters by drawing ellipses around them [4]. In Figure 7(b), we then see the separation of outliers (anomalous data points) from the normal clusters, in which these outliers are points which do not fall into any of the clusters formed [4]. The advantages of these techniques are that it is easy to find outliers when working with small-scaled datasets. Another advantage is that "bursty and isolated" [4] anomalies can be analyzed accurately. The drawbacks of these techniques include the fact that only continuous attributes (Examples listed in Chapter 3 Figures 1-5) are used for most of the proposed techniques. In NAD, an hypothesis is that "larger clusters are normal" [4] while smaller ones are attack or intrusions. It is challenging to evaluate these techniques without this hypothesis. The use of disproportionate measurements influences the detection quota negatively. The computational complexity can be quite high when compared to other NAD techniques as most techniques use both clustering and outlier detection [4].

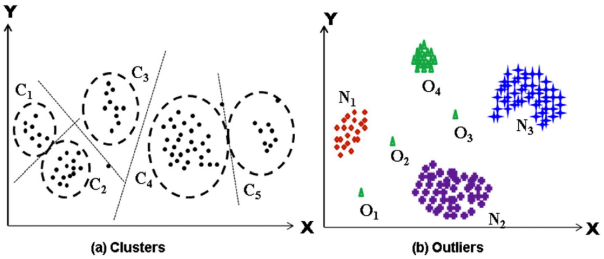


Figure 7: Clustering and outlier-based NAD from [4]

4.1.4 *Soft Computing*

Soft computing techniques are sufficient for NAD because sometimes it is impossible to find exact solutions [4]. Soft computing is divided into the following methods : (i) Genetic algorithm (GA), (ii) Artificial Neural Networks, (iii) Fuzzy Sets, (iv) Rough Sets and (v) Ant Colony algorithms and artificial immune systems [4].

(i) Genetic Algorithm (GA) is a population-based adaptive heuristic search for finding solutions to optimization and search problems based on the concept of biological evolution [4]. The approach is to evolve a population of possible solutions towards a better solution, which will in turn, result in better detection rates of anomalies. GA uses three main steps to determine the next generation: (a) selection, (b) crossover and (c) mutation. Each solution has a set of properties called chromosomes, which can be altered. Initially, selection involves using a fitness-based mechanism, where a sample of fitter solutions get picked from a huge population of random genes. The higher the fitness score a chromosome has, the better the chance it has to be selected. The selected solutions are then used for crossover. This is when a new type of chromosome is generated by exchanging random selected segments from the chosen chromosomes. The mutation alters existing chromosomes as the new type will now contain parts that cannot be found in the original ones. The process is then repeated until the best suitable solution is found. This solution must meet the criteria that was decided upon before the process and it will be optimal, such that successive iterations will not be improving the results. However, there are some limitations to this method, including the amount of time it may consume for the process [12] [7]. For example [7] uses GA to develop and improve rules for NAD. This example is further explained in Chapter 4.3.

(ii) Artificial Neural Networks (ANN) are inspired by recognition that our brain works in a completely different way compared computers. The brain performs certain computations (e.g. pattern recognition, motor control) much than the fastest computer. ANN is based on a collection of connected units, called neurons. The connection between two neurons is called a synapse and can transmit a signal to another neuron. Thus, in order to achieve a good performance, real neural networks utilize massive interconnections of neurons. Neural Networks learn from their environment by changing interconnection strenghts and synapse weights of the network [4] [6]. In the context of NAD, ANNs are used for data clustering, feature extraction and similarity detection, which is further detailed in [3].

(iii) Fuzzy network intrusion systems are used to determine whether malicious activity is taking place on a network using fuzzy rules. The system combines simple network traffic metrics with fuzzy rules to determine the probability of specific or general network anomalies. Once metrics are available, they are evaluated using a fuzzy set theoretic approach [4].

(iv) Rough set is a mathematical tool for feature extraction in a dataset generating explainable rules for intrusion detection [1].It is used when we do not have complete knowledge of the system. The mathematical framework of rough set theory enables modelling of relationships with a minimum set number of rules. For example, this method first extracts a minimum set of detection rules as the system generates sequences from the normal behaviour model during the execution of a process. With these rules, it then detects any abnormal operating status of the process and reports the abnormality as a possible intrusion. The benefits of using this technique are: (i) enables learning with small size training datasets and (ii) overall simplicity [3] [4] .

doi: 10.2313/NET-2017-09-1_08

(v) Ant colony optimization (ACO) and related algorithms are techniques for solving computational problems, which can be rephrased to search for optimal paths through graphs. The algorithms tries to mimic the behavior of ants searching for a path between their colony and a source of food. Artificial Immune Systems (AIS) represent a computational method inspired by the principles of the human immune system. The human immune system is proficient at performing anomaly detection [4]. ACO in this context is used for feature selection and network anomaly detection, which is further explained in [3].

The advantages of soft computing-based anomaly detection are that these systems show a very high amount of flexibility and adaptability [6]. Unsupervised learning, using competitive neural networks, is effective in data clustering, feature extraction and similarity detection. Drawbacks of these techniques are that they have a high resource consumption as well as high dependency on the hypothesis, about the behaviour accepted for the system. This means that the training of the systems become very difficult, if there is a lack of normal traffic data.[6]

4.1.5 *Knowledge-based NAD*

Knowledge-based methods are utilizing network or host events and check these against known attack patterns and predefined rule sets. [4] separates these methods into two different categories: (a) rule-based and expert system approaches and (b) ontology and logic-based approaches.

(i) The expert system is a rule-based system, without or with knowledge. This system matches rules against the current state of the system utilizing a rule engine. Depending on the outcome, it fires one or more rules [4]. A very popular example of a rule-based Network Intrusion Detection System is Snort [13], which is an open source intrusion prevention system capable of real-time traffic analysis and packet logging.

(ii) Ontology and logic-based approaches use expressive logic structure in real time to model attack signatures by integrating statistical properties and constrains [4].

The advantages of these techniques are flexibility, robustness and scalability. Additionally they have a high detection quota, if training datasets are available for both normal state and anomalies. Drawbacks are the costs and time consumption for the development of a high-quality knowledge. It is very challenging for these techniques to detect unknown anomalies [4].

4.1.6 *Combination learners*

Combination learners are combining multiple techniques and split into three different categories: (i) Ensemble-based methods, (ii) Fusion-based methods and (iii) Hybrid methods [4].

(i) The idea between ensemble-based methods is to consider various classifiers and combining them into one, which outperforms all of these. These techniques evaluate individually and combine them to reach a final verdict. Advantages of ensemble-based methods are that even though the individual classifiers are weak, the ensemble techniques still perform well by combining various classifiers. Another advantage is that they are usable on large-scaled datasets and the set of controlling parameters are extensive and can be easily adjusted. The drawbacks of these techniques are finding consistent performing classifiers from a pool of classifiers is challenging [4].

Features	Description
f_1	Number of TCP Flows per Minute
f_2	Number of UDP Flows per Minute
f_3	Number of ICMP Flows per Minute
f_4	Average Number of TCP Packets per Flow over 1 Minute
f_5	Average Number of UDP Packets per Flow over 1 Minute
f_6	Average Number of ICMP Packets per Flow over 1 Minute
f_7	Average Number of Bytes per TCP Flow over 1 Minute
f_8	Average Number of Bytes per UDP Flow over 1 Minute
f_9	Average Number of Bytes per ICMP Flow over 1 Minute
f_{10}	Average Number of Bytes per TCP Packet over 1 Minute
f_{11}	Average Number of Bytes per UDP Packet over 1 Minute
f_{12}	Average Number of Bytes per ICMP Packet over 1 Minute
f_{13}	Ratio of Number of flows to Bytes per Packet (TCP) over 1 Minute
f_{14}	Ratio of Number of flows to Bytes per Packet (UDP) over 1 Minute
f_{15}	Ratio of Number of flows to Bytes per Packet (ICMP) over 1 Minute

Figure 8: Flow-based features used in [10]

(ii) Fusion-based methods is trying to improve classification accuracy compared to individual decision-based techniques by merging data on (a) data level, (b) feature level and (c) decision level. Data fusion is efficient in terms of increasing timeliness of attack identification and helps reducing false alarm rates. Another advantage would be that decision level fusion have a high detection rate when given applicable training data. The downsides of these techniques are the high consumption of resources as well as the feature level fusion being a time consuming task [4].

(iii) Hybrid methods in [4] are combining various known methods trying to create a new system due to anomaly-based NIDS having a too high false positive rate as well as misuse-based NIDS, which only detects known intrusions. This approach differs from (i) as it also uses misuse-based NIDS. These methods benefit using features from both signature and anomaly-based network anomaly detection. This leads to being able to handle both known and unknown anomalies. The drawbacks of these techniques are that they cost a lot of resources and updating rules or profiles or signatures dynamically remain difficult [4].

4.2 NAD using CUSUM and EM Clustering

The following section shows an example of a system, which uses CUSUM, non-parametric CUmulative SUM, and EM, Expectation-Maximization, based clustering algorithm [10]. The features used in the survey are all flow-based packet features, which mainly are the number of flows, the average number of packets, bytes per flow and the average number of bytes per packet in a set time interval (Figure 8).

The following Figures 9 (EM) and 10 (CUSUM) show that EM outperforms CUSUM in almost every feature category, as the detection rate (DR) is significantly higher for nearly all features. Although, both techniques show a significant amount of false positive rate (FPR). This leads to the conclusion that despite being able to detect anomalies correctly with a decent rate, the system is buried by false alarms as the quota is the FPR for each feature is almost over 80 percent for each feature in both tests.

Features	Average DR (%)	Average FPR (%)	Ratio of Avg. DR to Avg. FPR
F1	39.83	81.84	0.487
F2	52.22	84.04	0.621
F3	32.25	84.14	0.383
F4	12.0	89.03	0.135
F5	51.8	85.74	0.604
F6	32.25	84.17	0.383
F7	3.2	82.92	0.0386
F8	49.26	84.19	0.585
F9	32.25	84.14	0.383
F10	6.81	86.71	0.0785
F11	0.0	0.0	0.0
F12	32.25	84.17	0.383
F13	8.57	94.59	0.0906
F14	52.41	83.59	0.627
F15	32.25	84.17	0.383

Figure 9: Performance of EM detector used in [10]

Features	Average DR (%)	Average FPR (%)	Ratio of Avg. DR to Avg. FPR
F1	11.04	80.43	0.137
F2	12.96	85.94	0.15
F3	1.6325	87.33	0.02
F4	4.9	84.44	0.058
F5	17.84	82.42	0.217
F6	7.23	95.5	0.757
F7	2.94	79.61	0.037
F8	33.57	78.18	0.429
F9	11.5	81.1	0.142
F10	1.4	94.87	0.015
F11	0.7	95.24	0.0074
F12	10.8	87.26	0.124
F13	6.015	77.18	0.078
F14	27.73	81.85	0.339
F15	12.87	86.12	0.15

Figure 10: Performance of CUSUM detector used in [10]

4.3 Rule-based NAD using GA

Table 1: Training data test results used in [7]

Type	Occurence	Correct Identif.	Incorrect Identif.	Reliability
Normal	5000	4460	540	89.2
Attack	5139	4864	275	94.64

Table 2: Test data test results used in [7]

Type	Occurence	Correct Identif.	Incorrect Identif.	Reliability
Normal	5040	4710	330	93.45
Attack	4958	4670	288	94.19

Table 3: Iterations relative to accuracy in [7]

S #	Iterations	Accuracy (%)	
		Training	Test
1	500	74	71
2	1000	81	79
3	1500	86	84
4	2000	93	91

This section shows another example of NAD. The technique proposed in this paper [7] uses genetic algorithms to establish rules for NAD. On this occasion a chromosome in an individual consist of genes matching attributes such as service, flags, super-user attempts and being logged in or not. The tests between training data (Table 1) and test data (Table 2) show promising results as well as a very low false alarm rate. Table 3 shows the comparison between different iterations and their accuracy after evolving the rule sets with GA. These results show that more iterations lead to a significant higher amount of accuracy. The rate of increase in accuracy decreases as the number of iterations increases. Thus, this concludes that there is a reasonable amount of iterations which should be made but further iterations at some point do not increase the accuracy of the result.

4.4 Evaluation

This section evaluates all the discussed techniques presented in this paper. The techniques share the common goal of detecting anomalies however work in a completely different way. Choosing which one to use depends on the resource consumption, robustness, false alarm rates and detection rate. Depending on the situation, the certain technique's adaptability will determine how much information one can learn from training datasets. Analysing the main advantages and disadvantages of each type that was discussed in Chapter 3 and 4, it will allow us to compare the different techniques.

Starting off with statistical-based NAD, these techniques are very susceptible to wrong parameters and hard to adjust. They are reliable when it comes to detection rate and do not rely on training data. However, this means that their only way of adapting is to change the parameters, which may prove to be difficult especially in balancing false positives and negatives. Classification-based techniques are highly adaptive, but rely heavily on training data and do therefore

have high detection rates for known anomalies only. Thus, it is challenging to work with classification-based NAD on high-dimensional data. Clustering and outlier based algorithms have the massive drawback since they can only work on continuous data, which restricts them from using data like the protocol or flags mentioned in Chapter 3 (Figure 1-4). Other than that, they can perform well with small-scaled datasets by accurately detecting anomalies. Soft computing methods have the advantage of high adaptability and flexibility. The only disadvantages for these techniques are the reliance on training datasets and the high resource consumption. Knowledge-based techniques profit from high flexibility, scalability and robustness. However, they are reliant on training datasets, which leads to a high detection rate for known anomalies only. It is also challenging to detect rare or unknown anomalies as well as the acquisition of high-quality training datasets. Combination learners are also very dependent on training data, have a high resource consumption and are difficult to adjust. On the other hand, depending on the type of technique used, it can detect both known and unknown anomalies.

5. CONCLUSION

We conclude that Network Anomaly Detection covers a wide variety of interesting features, which algorithms can work with. The algorithms itself show a huge diversity between them, ultimately ending up with one goal: anomaly detection on network data. Altogether it can be said that each method has its own advantages and drawbacks. Therefore, it is challenging to determine which is the best type of algorithm to work. The main reason is that it depends a lot on the user's goal and purpose, availability of the data as well as the amount of available resources.

6. REFERENCES

[1] A. O. Adetunmbi, S. O. Falaki, O. S. Adewale, and B. K. Alese. Network intrusion detection based on rough set and k-nearest neighbour. *International Journal of Computing and ICT Research*, 2(1):60–66, 2008.

[2] J. P. Anderson et al. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.

[3] D. K. Bhattacharyya and J. K. Kalita. *Network anomaly detection: A machine learning perspective*. CRC Press, 2013.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, Systems and Tools. *IEEE communications Surveys & Tutorials*, 16(1):303–336, 2014.

[5] Dennis Schirrmacher, heise.de. Available online at https://www.heise.de/security/meldung/Rekord-DDoS-Attacke-mit-1-1-Terabit-pro-Sekunde-gesichtet-3336494.html ; last accessed on 2017/07/02.

[6] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18–28, 2009.

[7] M. S. A. Khan. Rule based network intrusion detection using genetic algorithm. *International Journal of Computer Applications*, 18(8):26–29, 2011.

[8] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. SIAM, 2003.

[9] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 120–132. IEEE, 1999.

[10] W. Lu and H. Tong. Detecting network anomalies using cusum and em clustering. In *ISICA*, pages 297–308. Springer, 2009.

[11] M. Mahoney and P. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Recent advances in intrusion detection*, pages 220–237. Springer, 2003.

[12] T. Shon, Y. Kim, C. Lee, and J. Moon. A machine learning framework for network anomaly detection using svm and ga. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 176–183. IEEE, 2005.

[13] Snort. Available online at `http://snort.org`; last accessed on 2017/07/03.

[14] M. Thottan and C. Ji. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204, 2003.

State of the Art Analysis of Defense Techniques against Advanced Persistent Threats

Jeslin Thomas John
Supervisor: Dr. Holger Kinkelin
Seminar Future Internet SS2017
The Chair of Network Architectures and Services
Faculty for Informatics, Technical University of Munich
Email: jeslin.john@tum.de

ABSTRACT

Advanced Persistent Threats (APT) have become one of the most serious IT security issues in the recent times, owing to reasons of its complexity, duration and inability of proper tracking. The threat posed by APTs have turned out to be a major concern not just for IT Firms, but also for industrial establishments, governments, military organizations etc. Although different research proposals exist in the academia, many of those are not yet reflected in existing solutions in the market. The following paper attempts a state of the art analysis of the different works from academic research and commercial solutions available in the market, compares their benefits and consequences to arrive in to a taxonomic illustration of the defense mechanisms. In addition, the paper also proposes a novel approach for APT Defense based on theoretical computational methods.

Keywords

APT Detection, APT Defense, Cyber Attacks, Attack Pyramid, Attack Trees, Fractal Analysis, Machine Learning, Intrusion Kill Chains, Command & Control Communication

1. INTRODUCTION

The sophisticated, targeted, persistent and well planned cyber attacks targeting specific organizations, governments, military groups etc. are collectively referred as *Advanced Persistent Threats (APT)*. [10] These attacks are mostly complex, stealthy and generally involve multiple stages that span over a long period of time. This makes APTs tough to detect, defend and mitigate.[11] The stages of an APT process are quite easily mistaken to be independent events that occur in unrelated slots of time. In this context, the static attack detection techniques prove to be ineffective and blacklisting and malware signature based techniques fail.[5]This clearly marks that the traditional approaches are inefficient in handling the sophistications of APT threats and novel approaches for detection and mitigation are required. As a result, innovative and proactive methodologies, that can provide security with an insight to continuously monitor the systems under protection, detect vulnerabilities and security breaches are being sought for. This would aid in minimizing the impacts caused to the target system, in cases of an attack. Various research proposals discussed in this paper makes use of a wide range of approaches for APT detection, depending upon the nature of the system under consideration.

In the next section, a background of the APT attacks are briefed which aids in better analyzing the approaches discussed further. Section 3 lists out the major works from the research world while section 4 details on some of the prominent industrial solutions available. Finally, the author discusses his own thoughts for the proposal of an APT detection system, based on theoretical computational methods, that can lead to better detections and defense.

2. BACKGROUND KNOWLEDGE

APTs are well planned security attacks, aimed at exploring the known vulnerabilities of a system, to get unauthorized access, by overriding the security and defense mechanisms in place. The attack goal in most cases would be to infiltrate in to the system, learn the internal activity flows and access confidential information. This demands a great deal of passiveness in attack operations, to go on undetected for a greater extent of time. It is *Advanced*, as the attacker makes use of a complex mix of diverse attack methods targeted and adapted to the vulnerable system, *Persistent*, as the attack life cycle is lengthy and can span over a long period of time without being detected as the attacker slowly goes on to acquire control over the target system, *Threat*, as the attack can cause damage of intellectual property that incur loss of money and reputation.[2]

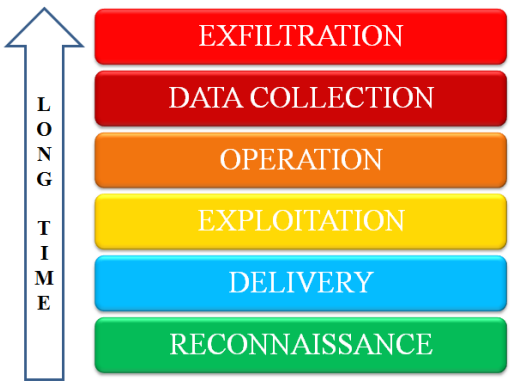


Figure 1: Typical stages of an APT

Figure 1 shows the typical stages of an APT with the possible attack strategies adopted in each stage described. APT attacks mostly begin by choosing an attack target, which normally would be a vulnerable spot inside the system that

doi: 10.2313/NET-2017-09-1_09

can let the attacker in. Once the target is fixed, the attacker normally loads a malware in to the secure network, which will establish a stealthy connection to a server outside the network. This helps him to learn more of the system and explore weaknesses by constant passive monitoring, minimizing attack footprints.This stealthy mode of communication is called a C&C, or *Command & Control* which taps the secure network details to an external system. [1]

3. ANALYSIS OF ACADEMIC PROPOSALS

A variety of proposals from the academic researches were analyzed to classify and compare the major approaches that are employed in the detection and defense of APTs. The benefits and demerits of these approaches were compared and contrasted to provide the reader a broader picture of the defense mechanisms and their applicability. As an outcome, a taxonomy of the various approaches was also generated, based on the underlying techniques used, which gives a neatly classified reference for further studies.

3.1 Detection using Fractal Dimension based Machine Learning Classification

The major idea behind most of the APT prediction methodologies is to identify some unique feature in the whole behavior of the system under consideration, and to track this uniqueness. The unique feature is then used to identify any possible deviations from intended behavior and efficient defense mechanisms are formulated. One major approach for APT defense using this unique property feature is the use of machine learning based techniques for fractal analysis.[1] Fractals are infinitely scaled and iterated abstract patterns often emulated in nature. Fractal analysis is a contemporary method of applying nontraditional mathematics, to patterns that defy understanding with traditional Euclidean concepts.[1]

The proposed system aims at representing the network level Command and Control (C&C) communications of the APT processes as fractal dimensions and implementing a fractal based machine learning algorithm, which compares it's output with a standard machine learning algorithm. Any significant deviation observed in the comparison can possibly suggest the presence of a valid APT attack.[1]

On a brief note, the defense system starts itself by extracting feature vectors after building a data set by combining the packet capture (PCAP)[15] files from various online sources. Noise reduction is performed on this data set and is presented as an input to an anomaly classification algorithm. The major algorithms employed for anomaly detection are the K- Nearest Neighbors and the Correlation -fractal dimension based algorithm. At any instance, one of these algorithms are chosen based on the intended performance specifications such as accuracy, sensitivity, specification or precision. The algorithm thereby predicts the possible valid inputs from the given data set that qualify for an APT.[1]

One successful implementation making use of this idea, based on TCP attributes, was proven to be efficient in the research set up.[1] In any case the accuracy of predictions always depends up on the quality of the training data that are fed in to these machine learning algorithms. So the data extraction and the prehistoric standard data must be clean and inclusive so as to produce the most accurate prediction results.

[1]https://imagej.nih.gov/ij/plugins/fraclac/FLHelp/Fractals.htm

3.2 Defense using System and Attack Intelligence

APT attacks are posing major threats not only to IT Infrastructure, but also to the Industrial control systems in critical functional domains such as oil and gas manufacturing, refineries and nuclear plants. The recent well known attacks in this domain includes the Stuxnet[14], which was targeted on destroying Iran's nuclear plans and Aurora[13], which was aimed at stealing Googles Intellectual property documents.

One work from academia aiming on detection of APTs in Industrial control systems introduces tools such as *Tofino*[12] and *Defender*[17] as the backbone. The use of *Tofino* enables Deep Package Inspection (DPI) with convenient and easy configuration steps, offering a standard network monitoring solution. It also runs in compatibility with the standard industrial control systems such as PLC(Programmable Logic Controllers), SCADA(Supervisory Control And Data Acquisition) and DCS(Distributed Control Systems)[4]

The detection approach monitor all the events that can occur in the system, records and projects the relevant items that may qualify for a sequence, in a succession of APT events. The recorded events are then matched with the already known attack behavior patterns and alarms are generated whenever a match is found. Though this seem to be a straight-forward and simple approach, the prediction can be fairly accurate and useful, provided the attack behavior database is well updated and inclusive.

As the detection approach is based on pattern matching between behaviors and events, techniques similar to formal method approaches for language recognition can be made use of. One possibility of such a technique will be the use of state machines that can predict an APT attack based on sequential event matching, with events as input symbols and patterns as transitions. One such implementation demonstrates a state table based approach enabling reduced space complexity, for detection of attacks similar to StuxNet.[4]

3.3 Detection using a Context-Based Detection Framework

APT threats can be also identified using a framework that can produce inferences based on context information. A conceptual model based on the *Attack Tree* concept is extended here to form an *Attack Pyramid*, which has the attack goal at the top of the pyramid and the event environments as the lateral planes. [2]

3.3.1 Attack Trees

An *Attack Tree* is a means to represent threats in a tree structure based on the work by Bruce Schneier[8] which originally uses the *Threat Tree* concept from Edward Amoroso[9]. The tree is constructed by positioning the goal of the attack as the root of the tree, and the various methods by which the goal can be reached, as children of the root. The child

doi: 10.2313/NET-2017-09-1_09

nodes in the tree can be connected by either *AND* rules, if both the nodes together should be present to reach the goal, or *OR* rules, if either of them can serve the purpose. On iteration, each of the child nodes are considered as a goal of the attack and subtrees are created with the goal as the root.

Attack trees help security administrators in an organization to create the hierarchy of vulnerable elements and the path of an attack, giving a big picture of the security architecture.

3.3.2 Attack Pyramid

The researches here modifies the attack tree concept to form a pyramid structure, called as an *Attack Pyramid*, which positions the attack goal as the root of the tree and the lateral planes as the environment where the attack evolves. The idea here is to map the APT stages to the planes of the pyramid namely physical plane, user plane, network plane, application plane and so on.[2]

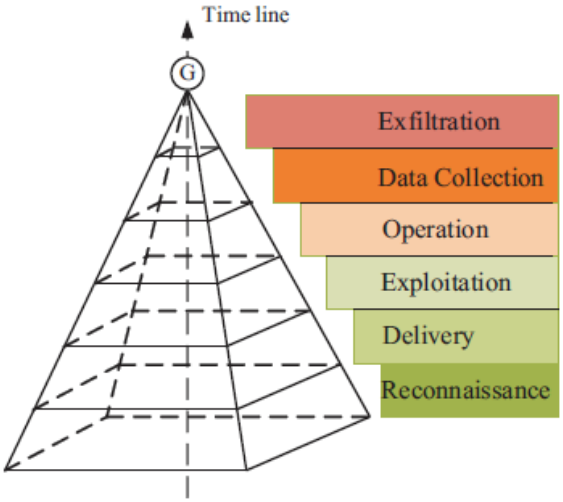


Figure 2: The Attack Pyramid. [2, Page 71]

The planes differ from system to system, depending up on the attack goal and the security architecture. Every attack in this kind of a model is viewed as a crawl along the planes to reach the goal, with multiple attack trees spanning the pyramid planes. The detection system aims at find relations between the occurrence of different events in the planes and building the attack crawl.[2]

3.3.3 Events

All the events that can occur in the system is classified here, inorder to associate events to system activities. Events can be *Candidate Events* which includes all logged events, *Suspicious Events* that are reported as an abnormal activity or *Attack Events* that are detected by the security tool as a valid attack activity.

The system records all the events that occur in the system and classify them to fit in to one of the event set. These events are then mapped to the pyramid planes using predefined mapping rules. The detection rules in this case can be based on signatures, policies or profiling. Mathematical correlations are then made by using correlation rules that finds the relations between independent events, to trace the attack path in the pyramid planes.[2]

The detection framework makes inferences taking in to consideration the context, the correlation rules, historical information, the confidence level and risk levels to arrive in to valid conclusions.

Figure 3 depicts the unfolded attack pyramid which clearly depicts the pyramid planes, and the corresponding APT stages. The crawl of a possible attack is marked with pointed arrows, as multiple attack trees to reach the goal G of the attack pyramid.

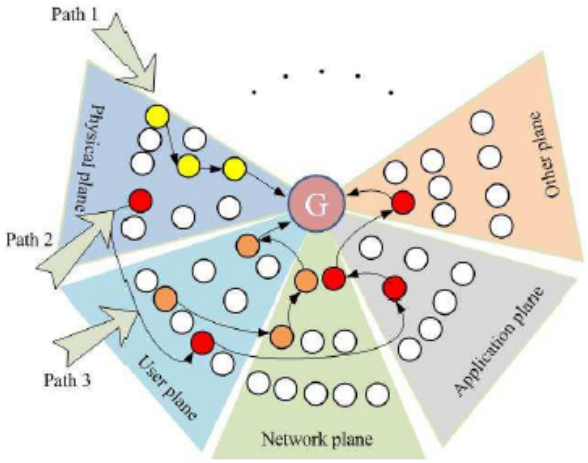


Figure 3: The unfolded attack pyramid. [2, Page 71]

3.4 Detection using Honeypots

Honeypots are computer systems devised as a decoy to mimic the real setup to deceive cyber attackers to detect, deflect and gain information on the methods employed by the attackers. The placement of honeypot traps in IT systems helps to enable an inexpensive detection of cyber attacks at an earlier stage including the ones missed by traditional intrusion detection systems.

The key idea in this kind of an APT detection approach is to tie up an alarming system to the honeypots devised, so that an early detection or warning on a intrusion can be made available. A properly configured honeytrap can be proven efficient in placing the attacker on a greater risk of getting identified, as a single mistake from the attacker side can alert a detection leading to an alarm. In addition, provided a low number of false positives, the defense team can timely devise counter measures while the attack advances.[3]

One successful implementation of such a honeytrap makes use of the *KFSensor*[3] tool, which is based on a Windows architecture and by writing a suitable Perl based alerting module.

doi: 10.2313/NET-2017-09-1_09

Table 1: Taxonomy of APT Defense Techniques

Defense Technique	Applicability	Implementation Available
Fractal Analysis	IT Systems	Yes, Psuedo Code
Context Based	IT Systems	Yes, Psuedo Code
System & Attack Intelligence	Industrial Control Systems	Yes, State Table Based
Honeypots	IT Systems	Yes, Perl Script
Intrusion Kill Chains	IT Systems	Yes, Using Hadoop FS
Distributed Computing Based	IT Systems	Yes, Using Hadoop

3.5 Detection of Multi- Staged attacks using Intrusion Kill Chains

Figure 4: The Intrusion Kill Chain.

Multi-staged cyber attacks are always hard to detect and defend due to their dynamic nature and the fairly unpredictable time frame that encompasses the course of events. While some APT detection methods aim at using the traditional malware detection mechanisms and blacklists, multi staged attacks usually tend to be a hard nut to crack. One notable research in this field proposes a system that makes use of the properties of an IKC(Intrusion Kill Chain)[20] that can suitably model the attack and enable early detection. IKC is a seven stage model that any attacker should inevitably follow to execute a successful APT attack[5]

The system evolves itself by proposing a layered security architecture that adapts to a multi staged attack model and then by collecting and analyzing the security events. The analysis of the security events includes logging the outputs from various sensors such as host intrusion detection system (HIDS), network intrusion detection system (NIDS), firewalls and so on, which are later processed by a Hadoop based log management module. The Hive queries on this distributed file system are analyzed by an Intelligence module which correlates it to a possible IKC. There also custom tools inside the system that performs code analysis and behavioral analysis for malware detection.[5]

The intelligence module predicts the possibility of an IKC primarily by analyzing the Hive query outputs from the HDFS system that contains the sensor logs. The analysis maps each of the suspicious event identified to one of the seven stages of the attack model formerly devised and then a defense plan is formulated which contains attack mechanisms and the possible defense strategies. Once the defense line is identified, it is mapped to an IKC phase, where from an IKC is rebuilt and the multistage attack is predicted.[5]

The research also demonstrates a real world implementation of the concept using commodity hardware running Hadoop clusters and with Apache Sqoop and MySQL.[5]

3.6 Taxonomy of APT Defense Methods

A straightforward classification of the APT defense methods available is depicted as in Table 1 above. The approaches are marked with information, if a valid implementation of the solution is available.

3.7 Analysis of approaches

The defense approach using honeypots are simple, straight forward and less resource intensive compared to the sophisticated ones proposed using fractal analysis, but is limited to the application domain with the dependency to the system behavior. Fractal analysis on the other hand proves to be better efficient as it involves thorough analysis of all network and host related data flows and mined data of relevant events and their consequences. Intrusion kill chains and Attack pyramids propose efficient means to track and alert on APTs alongwith valid predictions on the further progressions.

4. ANALYSIS OF INDUSTRIAL SOLUTIONS

In light of the various approaches discussed, various solutions available in the market for APT Defense are worth to

doi: 10.2313/NET-2017-09-1_09

Table 2: Comparison of Industrial Solutions

Feature	THOR	TrendMicro	Kaspersky	Symantec
Network Flow based	Yes	-	-	-
NIDS	-	-	-	Yes
Threat Intelligence	-	-	Yes	-
Automatic Correlation	-	-	Yes	-
Deep Package Analysis	-	Yes	-	-
Sandboxing	-	-	Yes	-

be analyzed. For the matter of analysis, couple of well known solutions were handpicked, compared and contrasted. It was observed by the author that only limited precise information on the product architecture and technology is provided by most of the vendors, which makes it hard to analyze the quality of the tool.

4.1 Defense Solutions based on Network Flow Analysis

Network flow analysis includes various approaches to collect and process network traffic, and all related data items that can be used for studying the behviour of the network, record activities and generate inferences. This evidence can be used for determining network features such as security, performance, integrity, capability etc. The following tools presented are based on flow analysis to detect discrepencies in a network.

4.1.1 THOR

One prominent implementation available in the market from BSK Consulting GMBH[16] that employs the concept of network flow based prediction is THOR.[16] The solution claims to implement deep system scanning for APT detection. THOR can be configured in a complete off-line operation mode, that leverages the flexibility by making it possible to merge logs from different network segments off-line. The solution makes use of the signatures maintained by security analysts and claims to have custom attack related patterns for enhanced detection capabilities. The solution also supports multiple output formats, ranging from text log to ArcSight CEF, which enables easier integration with SIEM systems. In addition, the solution claims to have an unknown malware detection feature, that uses a file scoring mechanism, based on attributes, contents and meta data.

THOR runs in three major use cases namely,

- Triage Sweep
- Single System Live Forensics
- Image Scan in Lab

The tool also can quarantine samples via network, using BiFrost[16] and can detect deleted elements by a disk surface scan, using DeepDive[16]

4.1.2 Symantec Endpoint APT protection

One of the well known players in the industry, Symantec uses an End point APT protection[2] using multi-layered approach. The strategy devised there is to use a combination of their products array to enable the APT protection, among which the Key player is an intrusion prevention system.

4.1.3 Kaspersky Security Operation Center

Kaspersky SOC from Kaspersky Labs claims to use a centralized threat monitoring approach using the so called *Security Operation Centers*[18], which is a team of cyber defense engineers and resources who can act upon the alarm of a security incident. In addition, the solution provides support for threat intelligence and automated correlation. The firm also provides a *Security Network* tool that is supposed to be an instant reactor to APT threats and a *Automatic Exploit Prevention Technology* with their Kaspersky lab protection solutions, that is supposed to block exploits in targeted attacks. This tool also supports standard white-listing modes which is aimed at reducing the attack surface.

4.2 Defense Solutions based on Deep package analysis (DPI)

4.2.1 TrendMicro DeepSecurity

One noteworthy solution found in the market employing DPI Analysis is *Trend Micro Deep Security*[19] which provides a *Deep Discovery Inspector and Advisor*[7] that claims to be equipped enough to filter malicious content by sandbox simulation of suspicious files, by means of a passive non-intrusive mechanism. The tool also provide destination analysis and communication fingerprinting for tracking C&C The standard here is to follow a Rule based heuristic analysis and perform a Deep packet inspection for protocol detection. To add on, the solution extends itself with a handful of threat scan engines for threat detection and makes use of correlation techniques.

The tool is claimed to be equipped with dedicated threat engines and multi-level co-relation rules, that aids in the detection of threats with minimal false positives. The sandbox analysis is done using a *Virtual Analyser* which provides a quarantine for safe explosion of the threat to perform an in depth forensic analysis. One necessary feature of these tools is the compatibility it should have with standard SIEM(Security Information and Event Management) consoles. SIEM systems are used to analyze secuirty events collected from various sensors residing in the network.This solution facilitates compatibility with standard SIEM platforms, enabling Enterprise wide threat management from a single SIEM console.[7]

[2]https://www.symantec.com/content/dam/symantec/docs/datasheets/atp-endpoint-en.pdf

doi: 10.2313/NET-2017-09-1_09

4.3 Other Solutions

One another solution available in market for APT Defense is the *WildFire*.[3] The solution claims to be performing Sandbox analysis for secure and controlled execution of the threats and DNS based intelligence to track any C&C activity that might reflect an APT process.

4.4 Analysis of industrial solutions

On analysis of available implementations, the gap between academic research and solutions is pretty wide. Most of the vendors are aiming to promote their business based on the strategy of selling their product ranges as a whole rather than showcasing a single product that can cater to the problem. As most of the solutions don't expose their internal implementations, its hard for a naive customer to assess the efficiency of each of them.

The proposal for a better industrial product swill be to use a hybrid system that encompasses the features of honey pots for early detection and misleading for systems which are regularly monitored by a security team, fractal analysis for IT infrastructures involving heavy traffic flows and globally distributed network by backing up with a distributed multi staged detection system, system and attack intelligence for industrial systems and intrusion kill chains clubbed with context behavior in case of middle-sized implementations.

5. PROPOSAL

As an inference from the analysis of existing solutions and the academic research, the author have a theoretical proposal for a better efficient self-learning defense system, which relies on the principle of artificial intelligence and context sensitive computational methods. The current research works its way on by identifying relevant events from the noise created by an APT attack and aims at correlating this to one of the identified stages in an APT life cycle. Later on, an alert is generated and the system administrators are informed of a possible APT attack. This doesn't appear to be highly effective, as the systems mostly generate independent alerts which needs manual intervention to correlate between events to recreate the attack flow.

An alternative approach would suggest the use of a mechanism similar to that of context sensitive automatons, to create an APT life cycle generator as a context sensitive machine with each of the possible APT cycles, as a language generated by the machine. On each step when an APT qualified event is encountered, a context sensitive transition is performed by the life cycle generator, with the event as an input symbol and the APT stage associated with the successful completion of the preceding event as the previous state of the machine. The idea here is to try and match every new event with an existing APT generator if a valid transition is present, or else to spawn a new machine with its current state corresponding to the event. A transition table is constructed from the input events and the stages, that will contain the permutation of all possible event occurrences from every stage, and the stage changes that can happen.

[3]https://www.paloaltonetworks.com/features/apt-prevention

The APT generator machine introduced above can be mathematically represented as follows :

M_{APT} = *(Q,* Σ *,* Γ *,* δ *,* q_0*, Z, F)* where

Q : Q is a finite set of APT life stages.

Σ : Σ is a finite set of relevant input events

Γ : Γ is a finite set called the stack alphabet.

δ : δ : (Q $\times$ ($\Sigma \cup \{\epsilon\}$) $\times$ Γ $\times$ Q $\times$ Γ^*) , is a finite subset of all possible decision moves.

$q_0 \in$ Q, is the start stage

Z$\in$ Γ is the initial stack symbol

F$\subseteq$ Q is the set of accepting stages.

On encountering a new event, the system will try to find a machine among the ones available to perform a possible transition that will lead to the creation of a portion in the APT life cycle.This approach will prove really efficient with APTs, as the time - variant, long spanning independent events can be easily correlated and further stages can be predicted with greater accuracy. The system will also train itself using a machine learning approach, to expand is transition table so as to match with the threat database.

6. CONCLUSION

Advanced Persistence Threats as discussed in this paper are mostly complex, stealthy and sophisticated. It generally involve multiple stages that span over a long period of time. Detection methods for these threats are not straight forward due to it's complexity and longer duration. Nevertheless, standard security measures based on blacklisting and malware signatures are ineffective. These situations naturally call in the need for innovative systems that enable continuous monitoring of the network and all associated interactions to the system. A handful of such approaches were analysed in this paper.

Different approaches exist in research for APT detection, but only a few have successful working implementations. A proposal would be a hybrid of many approaches, which would work better than just one. The industrial solutions available in market incorporate many of the research concepts described in this paper. In addition to the approaches analyzed, the author proposes a detection system based on formal methods which is efficient to identify and track associated events in an APT attack.

7. REFERENCES

[1] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens and Witold Kinsner: *Detecting Advanced Persistent Threats using Fractal Dimension based Machine Learning Classification*, In Proceedings of IWSPA'16, pages 64-69, , ACM, March 11 2016, New Orleans, LA, USA

[2] Paul Giura and Wei Wang: *A Context-Based Detection Framework for Advanced Persistent Threats*,

In Proceedings of the 2012 International Conference on Cyber Security, pages 69-74, IEEE, 2012

[3] Zainab Saud and Dr M Hasan Islam: *Towards Proactive Detection of Advanced Persistent Threat (APT) Attacks using Honeypots*, Islamabad, Pakistan, In Proceedings of SIN '15, The 8th International Conference on Security of Information and Networks, Pages 154-157, ACM 2015

[4] A. Redondo, Aitor Couce-Vieira and Siv Hilde Houmb: *Detection of Advanced Persistent Threats Using System and Attack Intelligence*, In Proceedings of EMERGING 2015 : The Seventh International Conference on Emerging Networks and Systems Intelligence, pages 91-94, IARIA, Hamar, Norway, 2015

[5] Parth Bhatt, Edgar Toshiro Yano and Dr. Per M. Gustavsson: *Towards a Framework to Detect Multi-Stage Advanced Persistent Threats Attacks*, In IEEE 8th International Symposium on Service Oriented System Engineering, April 2014

[6] Paul Giura and Wei Wang: *Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats*, In Proceedings of the ASE 2012, [Online]. `http://web2.research.att.com/export/sites/att_labs /techdocs/TD _101075.pdf`

[7] *Trend Micro DEEP DISCOVERY Data Sheet*, [Onine]. www.trendmicro.de/media/ds/deep-discovery-inspector-datasheet-en.pdf, Deep Discovery 3.2 NY, Trend Micro, 2012

[8] B. Schneier : *Attack Trees - Modeling Security Threats*, Dr. Dobbs Journal, December 1999

[9] E. G. Amoroso : *Fundamentals of computer security technology*, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[10] RSA : *RSA Security Brief: Mobilizing Intelligent Security Operations for Advanced Persistent Threats* , http://bit.ly/gaf8hj, February 2011.

[11] Vries, J.D. and Hoogstraaten H. and Berg, J.V.D. and Daskapan S : *Systems for Detecting Advanced Persistent Threats CyberSecurity*, 54-61, IEEE Computer Society 2012

[12] E. B. E. Schweigert and M. Thomas : *Securing ethernet/ip control systems using deep packet inspection firewall technology*, Tofino Security, 2014. [Online]. Available: https://odva.org/Portals/0/Library/Annual20Meeting202014/2014 ODVA Conference Byres Schweigert Thomas Securing EtherNetIP with DPI FINAL.pdf.

[13] M. Zeller, *Myth or reality-does the aurora vulnerability pose a risk to my generator?*, in Protective Relay Engineers, 2011 64th Annual Conference for. IEEE, 2011, pp. 130-136.

[14] M. Kenney, *Cyber-terrorism in a post-stuxnet world*, Orbis, vol. 59, no. 1, 2015, pp. 111-128.

[15] PREDICT. (2009) *DARPA Scalable Network Monitoring(SNM) Program Traffic.*

[16] *THOR*, [Online]. Available: https://www.bsk-consulting.de/apt-scanner-thor/

[17] M. Brian : *Industrial defender solutions*, Lockheed Martins, 1996, retrieved: May, 2015. [Online]. Available: http://www.wurldtech.com/.

[18] *Kaspersky SOC* [Online]. https://www.kaspersky.com/enterprise-security

[19] *TrendMicro DeepSecurity*, https://www.trendmicro.com/en_ca/business/products/network/deep discovery.html

[20] Hutchins Eric M., Cloppert Michael J., Amin Rohan M, *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*, ICIW2011

Strategies for Malware in Cyber Conflicts

Vanessa Robl
Supervisor: Dr. Heiko Niedermayer
Seminar Future Internet SS2017
Chair of Network Architectures and Services
TUM Department of Informatics, Technical University of Munich
E-Mail: robl@in.tum.de

ABSTRACT

Over the course of the past few decades, the advancement of technology went farther and farther, reaching into nearly every corner of our lives. Information, or the deliberate concealment of it, can have an immense impact on the outcome of a volatile situation. Especially with regard to warfare the global interconnectivity of business, government, and infrastructure combined with a dependence on IT systems in all aspects of society brought forth a multitude of application possibilities which have never existed before. Conflicts can now partly be fought behind a computer screen. Lacking technological progress or the effort of highly skilled individuals can turn tables, providing major advantages or disadvantages to either participating side. In this paper, the term Cyber Conflict in association with Advanced Persistent Threats will be discussed. Thereby, the focus will lie on tools and strategies that are commonly used, as well as on methods on how to detect, defend from or distribute exploits.

Keywords

Cyber Conflict, Advanced Persistent Threats, Exploits

1. INTRODUCTION

"Cyber attacks include the unintentional or unauthorized access, use, manipulation, interruption or destruction (via electronic means) of electronic information and/or the electronic and physical infrastructure used to process, communicate and/or store that information. The severity of the cyber attack determines the appropriate level of response and/or mitigation measures: i.e., cyber security."[1]

With the introduction of personal computers to homes and businesses in the 1970s, the foundation was laid for the development of the enormous global network of interconnected autonomous networks we know today as the Internet. It became a tool offering great opportunities, but at the same time it came with a vast amount of risks. Where there were people to strive for education, knowledge or an open market there were also those to operate for their personal gain, gathering information, finding loop holes, dismantling organized structures. In general, networks have grown unbelievably large. Most command and control systems are connected to the Global Information Grid (GIG) or have embedded chips, making them vulnerable. The bigger the network, the higher the risk for a successful intrusion since it is harder to cover up such a huge attack surface. Additionally, the more influential or valuable the data within the system, the higher the temptation to sneak in and steal the data. Although the technological advance brought with it enormous and radical changes and benefits to our society it, at the same time, entails more and more risks.

To name only a few threats, there is malicious software like Trojans, Viruses, and Worms which can infect a computer system as well as Denial of Service attacks or Phishing. All of them have more or less the same goal to compromise the target system and benefit the attacker. The victims of such attacks are regular users as well as companies or even governments.[1][2]

Within this work, we will have a look at the agendas both attackers and defenders have as well as at strategies and tools that are used to achieve those.

2. DEFINITION AND TERMS

Before we can dive into the analysis of strategies regarding attack and defense of system vulnerabilities, we first need to establish some terms and definitions that will be used in the scope of this work.

2.1 Cyber Conflict[3]

Cyber Conflict or Cyber Warfare is a term defined as *"a tense situation between and/or among nation-states and/or organized groups where unwelcome cyber attacks result in retaliation."*[4] Involved are two or more opposing parties which can be made up of state-actors, companies, criminal organizations, hacktivists or similar. Strategies include defense against incoming attacks on infrastructure, economic disruption or loss of vital information while at the same time attacking the opponent in the hopes of causing damage, spreading confusion, using obfuscation to cloud movements or gaining valuable knowledge.

Cyberwar and conventional warfare in itself arenâĂŹt that different, both use the same means to achieve a strategic objective: attacking and espionage. Attacks are aimed to cause immediate damage or disruption to vital points within the enemies area of operation while espionage is a used to gaining an advantage by knowing more about the enemy than vice versa. Strategies and goals in both are quite similar as well when looking at the big picture. Despite these similarities, there is one glaring difference: the low barrier of entry. Cyber weapons are cheap and do not require significant infrastructure, financing or physical space for development and assembly. To deliver a potentially devastating attack on a city or even a whole country, the only tools needed are a computer, an Internet connection, and knowledge. In addi-

doi: 10.2313/NET-2017-09-1_10

tion, detecting or defending against a cyber weapon can be extremely difficult.

2.2 Advanced Persistent Threat[5]

Advanced Persistent Threats (APTs) are a special type of sophisticated and well-planned attack on a computer system, compromising it stealthily with the goal of staying undetected for as long as possible while continuously sending gathered data back to the attacker. Generally it can be organized into six successive phases:

Phase 1: Reconnaissance & Weaponization

- information about target is collected
- vulnerabilities are identified
- multiple attack patterns are prepared to be able to adapt to different cases

Phase 2: Delivery

- delivery of the exploit via the identified vulnerability
- direct delivery: attackers send exploits directly via various techniques (e.g. malicious email, drive-by downloads)
- indirect delivery: a trusted third party is compromised and the exploit is implanted by using them

Phase 3: Initial Intrusion

- exploit is successfully executed
- typically, backdoor malware is installed, granting access to the attacked system
- also possible: login credentials of the target have been obtained
- goal: establish a foothold in the compromised network

Phase 4: Command & Control

- taking control of the target network
- enabling further exploitation
- evading detection through the use of various methods for obfuscation

Phase 5: Lateral Movement

- discovering and collecting data
- internal mapping of the network
- taking control of additional (neighboring) systems
- usually longest lasting phase
- goal: getting large amounts of data over longest amount of time while running low to avoid detection

Phase 6: Data Exfiltration

- extracting the gathered data securely from the target system to a neutral system
- often: compression and encryption of the data
- hiding transmission

APTs are key tools in serious cyber conflicts. They are complex, tailored very carefully onto a specific target which has been chosen regarding its tactical and informational value. A high amount of skill, knowledge, and time is needed, but the gain in case the attack is successful can be tremendous. In a conflict, it is they ideal weapon, offering the possibility of high impact while having anonymity, thus, making it a dangerous and serious threat. For that reason, we will take a closer look at APTs in the scope of this work.
This paper will focus on the strategies and tools used in Phase 3: Initial Intrusion and Phase 5: Lateral Movement.

2.3 Exploits[6][7]

An exploit is a program specialized on taking advantage of weaknesses and vulnerabilities in auxiliary or application programs to gain access to systems and compromise them with the goal of obtaining advanced privileges like administrator rights.
There are various kinds of exploits, for example:

- **Local exploits:** can be activated by opening seemingly harmless files if the program with which it is opened has been compromised using a security flaw within it
- **Remote exploits:** an attack using manipulated data packets or specific bitstreams on weak spots in the network software
- **DoS exploits:** purposefully overload an application until it stops working
- **Zero day exploits:** term used for an attack using a yet undetected security loophole, making the developer aware of the vulnerability once he notices the intrusion

Protection from exploits starts with the careful programming of applications with the goal of having close to zero vulnerabilities. In addition, intrusion detection systems and intrusion prevention systems can help in finding and maybe even intercepting incoming attacks.

2.4 Persistence and Stealth[8]

Stealth and persistence are terms that will be used in a very specific way in the following context:
"The Stealth of a resource is the probability that if you use it now it will still be usable in the next time period. The Persistence of a resource is the probability that if you refrain

doi: 10.2313/NET-2017-09-1_10

from using it now, it will still be usable in the next time period."[8]
Stealth is designed to evade detection during transit, execution or when a program is at rest, whereas persistence is drafted to stay within an affected system or network beyond the termination of the delivery mechanism that was used.
High-level Advanced Persistent Threats are modeled in a way as to maximize the value of these afore-mentioned resources. The act of balancing the two factors to reach that maximum is a challenging task involving deep knowledge, calculation, and foresight to be able to assess the optimal time to strike. Thereby, the stakes and gains have to be taken into consideration. These values can not be calculated accurately or with certainty, since one can only assume what the situation, for example between two competing nation-states, might be in the future.
Formally, stealth and persistence can be depcited as equations (taken from Ref. [8]):

$$Stealth = Pr(resource\ survives|use\ it) \quad (1)$$

$$Persistence = Pr(resource\ survives|not\ use\ it) \quad (2)$$

Benchmarks for stealth and persistence are, for example, that the average duration of an attack based on a zero-day-exploit is 312 days (stealth) and that only three to five percent of vulnerabilities in popular browsers could be rediscovered within a three-years period (persistence). Both variables, of course, are also dependent on the state of the target system. Against a defender that keeps his system up-to-date and is overall well-protected, the stealth and persistence values will be lower than against one who is more lenient. Overall, the best time to strike is, when the gain is high enough to counterbalance the risk involved.

3. GOALS IN CYBER CONFLICT

In today's world a large amount of critical areas depend on large networks, partly using the Internet excessively: agriculture and food, banking and finance, communication, critical manufacturing, emergency services, energy, healthcare, nuclear reactors, transportation systems, and water to name just a few. A successful attack on any of these could potentially have disastrous consequences.[2]
There can be various motivations for an attack (taken from Ref. [12]):

- **Political:** protest against political action; protest against laws or public documents; protest against acts related to physical violence; espionage
- **Economical:** personal or corporate financial greed; espionage
- **Socio-cultural:** competition between groups over diverging goals, scarce resources; ethnic conflict

Being successful in a cyber conflict depends on two things: means and vulnerability. That means, the outcome is heavily influenced by the people, tools, and cyber weapons (means) available to each side and by the the extent to which either party uses the Internet and networks within their economy and military (vulnerability). The overall goal is to have higher educated employees than the enemy equipped with a more extensive and sophisticated set of tools and carefully developed cyber weaponry. This, of course, is only efficient if the enemy country is at a state of development where an attack directed towards networks will have an actual impact. For example, it would not make sense to order an attack on water and electricity distribution facilities aimed at a rural village with no water supply lines or electrical lead. Too big of a difference in technological development can make the use of cyber attacks obsolete.[13]
So far we looked at overall sources of motivation and general goals. For software or tools, there exist different kinds of goals depending on the purpose and the desired effect which can roughly be divided into Stealth and Persistence and Maximum Damage.
Majorly used in espionage, Stealth and Persistence based attacks aim at quietly infiltrating a system and staying there undetected as long as possible but with a reasonably well-established access to valuable data or essential settings. The goal, thereby, is, of course, to gather a large amount of usable information or to manipulate the system gradually and stealthily, reutilizing the same routine again and again. APTs can be classified as Stealth and Persistence attacks.
A variation of that which uses both aspects, Stealth and Perception as well as Maximum Damage, is planting a program into a system to disrupt or destroy it effectively at the optimal moment. This includes staying undetected until the attacker decides to start it. The more time passes, the higher the risk that either the malicious program itself or the attackers backdoor access for triggering it will be found. Also, if the attacking party miscalculates, and launches it either too early or too late, the desired effect might have way less impact or none at all.
For the Maximum Damage approach it is not necessarily important if the used tool or weapon can not be reutilized in other attacks, be it because it was detected and made public or because after detection there were countermeasures taken to prevent a similar or the same attack. After successful intrusion, the used tool will try to do the highest amount of damage possible without regards to stealth or persistence.
In comparison to attacking, when being hit by an attack or intrusion, the system recovery often is the top priority. By doing that, evidence necessary to determine how the systems was compromised is lost. Thus, the goal for a defending party is to do forensic work before a reload or it will be impossible to determine where the attack came from and how to develop countermeasures to prevent it from happening again. Generally, of course, the goal for a defending party is to prevent any kind of attack from happening in the first place by having strict security standards in place.[2]

4. TOOLS FOR CYBER CONFLICT

With the advance in technology it became impossible to rely solely on ones own skill in programming to mount a successful attack. There's a variety of tools available which are not necessarily meant for this abuse but, nevertheless, serve a much-needed purpose in aiding offenders. They can be categorized as follows:

- Reconnaissance tools
- Scanning tools

doi: 10.2313/NET-2017-09-1_10

- Access and escalation tools
- Sustainment tools
- Obfuscation tools
- Exfiltration tools
- Assault tools

A large amount of these applications are open source and/or freeware and are developed further and kept up-to-date constantly. In the following, we are going to look at each of the listed varieties, although we will mainly focus on those that might be useful for the before-mentioned APT-phases 3 and 5: Initial Intrusion and Lateral Movement. Every category will have the respective phase in brackets.[2][9][10]

Reconnaissance tools (Phase 1)
Reconnaissance Tools are used to gather information by, for example, accessing public websites, looking up Domain Name Server (DNS) records or collecting metadata from documents. For that, one can use automated data mining or search engines to filter for certain keywords. There are some search engines that can retrieve information even from the so-called "Deep Web" which normally is not accessed by the standard search engine.
An exemplary tool to find out information about domain names all over the world is textitWhois. It returns contact information which, in addition to the name server, may include the physical address, phone number, and contact name from someone linked to the domain if they are not hidden by a proxy. There is also the possibility to run a query for the actual IP address in case it is already known. If not, given the name server, one can use the command line queries *nslookup* or *dig* (list of all dns entries in a certain domain) to retrieve it. With this basic information plus perhaps additional auxiliary tools, it is possible to dig deeper and gather even more information.
Another approach on getting to know more about the target system is the use of metadata. Metadata is, so to speak, data about data. Looking at a regular file this might be, for example, user names who edited the file, paths where the file has been stored, coordinates of where a certain picture was taken, image thumbnails, and many more. Applications that aid users in finding this information are, among others, *Metagoofil* and *Exiftool*.
The best defense against tools such as those is to restrict as many sources of information as reasonably possible. Regarding the DNS server, one can deny zone transfer to unknown machines or use proxies to replace the actual data. Removing metadata can be done using the same tools as mentioned above. Of course, there will always be a certain amount of data that can't be hidden away completely, but limiting that amount is a first step to make a system more secure.[2][9][10]

Scanning tools (Phase 1)
The purpose of scanning tools is to find information about the system environment and the system itself in detail. They are positioned in a way to either have direct network access, or to be able to listen to network traffic, especially if the target system is connected to the Internet. Scanners, for example, try to map networks, scan ports, and determine the operating system a target host is running on.
Applications like *Nmap* and *Nessus* can be used to ping IPs, detect vulnerabilities, fingerprint operating systems, run traceroutes, scan ports, and much more. Some features involve detection avoidance, changing the speed of processes, and changing the communication method.
To defend a system from such tools is difficult. For the scanners to not pick up any information, there has to be no detectable data leakage at all. Traffic should always be encrypted and the usage of standard ports (like port 22 for SSH) should be avoided. Proper implementation of firewalls and the use of reverse proxies which limit the number of exposed ports and are meant to obfuscate the underlying system as well as analyze incoming requests can help as well.[2][9][10]

Access and escalation tools (Phase 3 to 5)
Access and escalation tools, as the name suggests, focus on gaining access to a system and, then, taking control of it and trying to expand this control. For this, a vast variety of applications is available. Some of the most commonly used ones are: *Hydra*, *John the Ripper*, *Metasploit*, and *Canvas*.
Both *Hydra* and *John the Ripper* are password attackers. They aim to find a working username and password combination for a target system, eg a webpage or a database, to gain easy access. While *Hydra* works through lists of possible passwords, similar to a limited brute-force-attack, *John the Ripper* attempts to recover the original password from an acquired hash.
In contrast to the afore-mentioned, rather basic, attack tools, the *Metasploit Project* operates in a more advanced fashion. It effectively aids an attacker in using exploits to invade a system. With a built-in list of usable bugs and exploits it checks if the target system is vulnerable to any of them and hands the user a selection of possibilities. The chosen method together with a configured payload that will be executed on the target system is encoded and, then, executed. Thereby, it is indispensable to have a certain degree of knowledge about the target which can be obtained by using one of the above-mentioned Reconnaissance or scanning tools. Similar to *Metasploit*, *Canvas* is based on using a library of exploits and payloads to compromise a system as well. Although its library is not quite as extensive as others, it is updated far more regularly, keeping up with the fast moving changes in the cyber environment.
A password policy with strong passwords and regular changes as well as patching and system hardening can keep tools such as this at bay to a certain extent. In regards to passwords: They should be at least 8 digits long, have at least one number and one symbol, and should contain both uppercase and lowercase letters. In addition, they should be changed around every three months. To defend against exploit abuses, one should always patch and update the system as fast as possible. In addition, it is beneficial to disable all ports, services, accounts, etc that are not absolutely necessary for the system to function. Taking these steps already has a great impact on the possible ways an attacker can gain access.[2][9][10]

Sustainment tools (Phase 3)
Once the desired level of infiltration has been reached after the first breach, an attacker needs to make sure that he can continue accessing the system since it may very well be possible that the vulnerability that has been used will be fixed

by a patch or similar measures in the future.
The easiest and, at times, most effective way to ensure that, is to create an account with the necessary rights for ones own use. This can be accomplished by simple system commands such as *useradd* or *netuser*. Nevertheless, there are better and less detectable methods, for example, installing a backdoor. A programmer with enough background knowledge can easily write such code by himself, but there is also a list of web-based methods found in the *Web Backdoor Compilation*. Additionally, for those who do not want to program themselves, there is a tool available called *Netcat* which will create a listening port granting access to a shell on the system. Using some tweaks, tactical renaming of the tool, and the careful selection of the used ports can minimize the risk of detection.
The defense against such actions can be divided into two parts. Firstly, by hardening the system as best as possible and restricting incoming and outgoing traffic we can make the insertion of backdoors difficult from the start. In addition, with the help of applications such as *powerbroker* or *Cisco Security Agent (CSA)* one can lock down administrative access to the system as a preventive measure against unwanted installations. Secondly, close monitoring of accounts, system accesses, open ports, and similar instances can help finding already installed backdoors. Since this is an immensely time-consuming task, most will not observe their system in such a way, thus, well-hidden installations may never be found.[2][9][10]

Obfuscation tools (Phase 4 and 5)
In order to stay hidden and undetected while covering their tracks at the same time, attackers need obfuscation tools. Generally, there are three main concerns.
Firstly, both the logical and physical location needs to be obscured. To do that, the use of proxies is common. Tools such as *The Onion Router (TOR)*, *Bitblinder* or *Perfect Dark* provide this service, offering anonymity of communication. Similarly, Virtual Private Networks (VPNs) or compromised private systems can be utilized as manual proxy.
Secondly, it is important for an attacker to eliminate all log files generated by his actions within the network to cover his tracks from future investigations or system administrators. With administration rights, most logs can easily be manipulated using simple text editors. If that approach fails, one can try to erase them entirely or overwrite them with self-generated events. In the worst case, if detection has to be avoided at all costs, extreme measures such as the use of assault tools to destroy or disrupt the system entirely have to be taken.
Thirdly, in case an intruder changed or created files and wants to hide those, there are different kinds of approaches. For example, one can rename the files and hide them within a list of similarly named files or use tools like *slacker* to hide them in places which are not easily accessible to users. With the help of the *NTFS file system* it is even possible to place data in Alternate Data Streams (ADS), storage areas originally intended for metadata. One more thing that has to be taken into account is to adapt the timestamp, so that it won't stand out from regular files. They are easily manipulated by either simple command line queries or a tool like *Timestomp*.
Defense against obfuscation tactics is almost entirely based on reaction instead of prevention. Without enormous resources or tactically placed layers it is very difficult to backtrack activities to the original location if the proxy is set up sufficiently. In certain cases, the application of tools that specialize in root kit detection can be helpful regarding preemptive measures. In contrast to that, defending against file and log manipulation is a bit easier. There are many measures that can be put in place, for example utilizing a tool called *Tripwire* to monitor for changes in realtime and alerting the defender when something seems out of place. Securing log files can be done by regularly storing them on a remote server, making them difficult to reach.[2][9][10]

Exfiltration tools (Phase 6)
After compromising a system and gathering data, it's necessary to have some way of retrieving all this information and putting it somewhere safe and more easily accessible. For that, exfiltration tools are needed. This can involve either physically transporting the data, using unfiltered protocols or hiding it through encryption.
For physical exfiltration, any storage media (memory card, USB stick, etc) can be used. With the progress in technology resulting in more and more disk space on smaller and smaller devices transporting them undetected is getting easy.
Another way to hide data effectively would be the use of encryption or steganography tools such as *TrueCrypt*, *Puff* or *OutGuess*. With them, collected information can be hidden from or at least made unreadable for a third party.
A third option could be to use the protocols that are already in place. It is, for example, possible, to send the encrypted and fragmented data via email. As an alternative, one can also use tools such as *OzymanDNS* to transmit the information via various protocols, such as HTTP or DHCP.
Unless the environment in which the system is located is restricted and diligently monitored or Data Loss Prevention (DLP) tools are used, it is extremely difficult to prevent anyone from exfiltrating data.[2][9][10]

Assault tools
Assault tools can not be assigned to an APT phase since the intention of APTs is to stay undetected for a long period of time while stealing data continuously instead of causing damage to the system. Nevertheless, they are an important part of Cyber Conflict. There is a vast amount of tools and methods available to harm, manipulate or temporarily disable a target. They can be related to both, software and hardware.
Once an attacker has administrative rights on a system, it is extremely difficult to prevent him from intercepting the regular workflow. Thus, the only effective way to defend against such an assault is to prevent attackers from gaining administration privileges in the first place. In some cases, there are users with admin rights who only need them for a limited amount of applications. Here, restricting the rights and assigning them solely for the purpose of executing those programs can secure the system further.
Affecting the software can, for example, be done by tampering with the system resources. Although easily detectable, it is a foolproof way to disturb a system. Using simple commands to fill up free space on the disk is only one way to make a computer unusable. Resources such as memory, CPU, and hard disk can be abused to run the attackers own implanted process, thus, preventing regular processes from working correctly. A less obvious, but also destruc-

doi: 10.2313/NET-2017-09-1_10

tive method involves the deliberate manipulation of the system environment. Many applications which depend on very specific settings can become unusable if one makes small changes. One of these delicate variables is time since a huge amount of programs depend on this variable to be consistent and correct over an extended period of time. There are various other small changes that can be made with an equally destructive result.
Methods to attack the hardware are generally a bit more complex. One way could be to rewrite Read Only Memory (ROM) modules which often contain firmware regulating how certain pieces of hardware function or communicate with each other. An easier way of attacking would be to tamper with the driver files, thereby, directly messing with the software that communicates with the hardware. This is only a temporary disruption, though, since it works only until the affected driver is reinstalled. There are many ways for attacking the hardware, but one of the most significant ones would be to interfere with the Supervisory Control and Data Acquisition (SCADA) system.[2][9][10]

To secure a system in general, there are various standard tools that should be used in a preventive measure such as firewalls, real time anti-virus protection, anti root kits, monitor tools, code review tools, and many more. Without basic protective programs, users basically open the doors to attacks on their systems. Of course, using such tools for defense doesn't give one hundred percent security, but by regularly scanning the system for programs with unusual behaviour or warning the user when an unauthorized download is happening, for example, can filter out at least some of the threats.

Attacking and defending with tools in a real cyber conflict can look like like the following:
In September 1998 the Electronic Disturbance Theater (EDT) attacked the Pentagon, trying to achieve a Denial-of-Service. Thereby, EDT developed malicious code and browser addins that reloaded a page over and over, resulting in a shutdown of services if enough people gather on the same target website (now known as *FloodNet* software). The defenders, as a response, programmed their websites in a way that, if a *FloodNet* attack was detected, the site would open a new window with every reload, eventually overloading the attacking system, causing it to shut-down and, thus, preventing the attack.[11]

5. STRATEGIES USED IN CYBER CONFLICT

Strategy as defined in Ref. [14] is the act of *"managing context for continuing advantage according to policy"* building on a systematic combination of goals and objectives, resources and capabilities, and ways to accomplish these goals and objectives.

5.1 Attack Strategies

Given the current stage of technology a strategic cyber attack by itself is not likely to be decisive for war. With a possibly uncontrollable blowback effect, parties involved in a conflict run a high risk of being hit back with the same attack they launched or with another, similarly disruptive one, unless the level of advancement in technology is vastly different. Generally though, the states that are most likely to develop the technological know-how on how to create and use a cyber weapon are also the most dependent on their own network infrastructure. Therefore, instead of launching direct attacks, using cyberwarfare capabilities on an operational level, as aid for regular troop movement and war strategy, might be more feasible. Nevertheless, cyber weapons can not, with some rare exceptions, destroy actual equipment permanently. The damage it can do is to confuse or interfere, to disable systems temporarily or to delay and obstruct certain processes.[13]
Based on this observation, reasonable objectives for a strategic cyber attack can be espionage, propaganda, Denial-of-Service (DoS), data modification or infrastructure manipulation.

Espionage
Cyber espionage is an expansion of the traditional effort to collect information on the enemies secrets, intentions, and capabilities. This includes the search for classified, personal or corporate data, intellectual property, proprietary information and patents as well as results from research and development projects. The gain in targeting sensitive information can possibly be very high. Stealing data can be done anonymously and remotely from all over the world which makes it a great strategic tool.[13][15]

Propaganda
Strategically placed propaganda can have a huge impact on peoples awareness and opinion on certain facts or situations. Using the amplification power of the Internet where digital data can be copied and sent instantly, a message can be spread within a short amount of time anywhere in the world. Casting doubt or spreading fear can have a huge impact on the pressure a government is exposed to, possibly even forcing it to behave differently to ensure the peoples support.[15]

Denial-of-Service
DoS is commonly used to temporarily make certain services or systems unavailable by flooding it with data or requests until it is unable to process all the data and simply shuts down. Other variations include the physical destruction of hardware or the use of electromagnetic interference which destroys unprotected electronics via current or voltage surges. From this, various strategical advantages can be gained. Depending on the targeted system and its use, there could be a loss of communication, a permanent loss of data, temporary blindness regarding surveillance or movements, loss of control over remotely controlled entities, and many more.[13][15]

Data modification
Corrupting data with a stealthy and undetected cyber attack can result in critical errors or wrong decisions made by the enemy based on their trust in the integrity of the information stored in their own network. Data modification can vary from implanting ones own propaganda or misinformation in a neutral or adverse website to the compromising of advanced weapons or command-and-control systems.[2][15]

Infrastructure manipulation
As mentioned before, the technological advances have given birth to highly vulnerable critical infrastructure connected to the Internet but without proper protection due to insuf-

doi: 10.2313/NET-2017-09-1_10

ficient computing resources or hardware restrictions. Some are dependent on realtime or automatic responses, making it more difficult to replace a failing systems with a human. A successful strike against, for example, an electricity plant or the traffic light system could lead to severe repercussions.[15]

Even though cyber attacks are not as obviously destructive as, for example, a bomb, when used correctly they can make a huge difference and give strategical advantages. In a conflict with more or less equipollent sides, it can turn the scale.

5.2 Defense mechanisms

Building up a decent and solid defense against attackers is a strategic challenge as technology and new software is developed so quickly that it is impossible for any organization to keep up-to-date with all of them. It is simply have too much ground to cover. An attacker only has to succeed once, whereas a defender has to constantly search for loopholes and improve, update, and maintain the overall system. In addition, it has to be guarded against insider threats and mobile devices that migrate in and out of the work environment.[2][15]
Nevertheless, defenders can make the most out of their situation. They have the 'home-field advantage' and with administration rights throughout the network they can change hardware and software configurations to their liking making their system unique and, thus, harder to crack since the standard vulnerabilities may not be there anymore. Knowledge is key. If a defender can limit the amount of information a reconnaissance or scanner tool can gather, it is the first and possibly most important step towards a more secure system. Attackers should need to work hard to gain even the slightest bit of insight while at the same time doubting if their obtained information is even correct or not. Defenses should be designed on the assumption that there is a breach in the network at all given times. Improving your own ability to collect, evaluate, and transmit digital evidence of attempted attacks or general traffic is a very good short-term cyber defense goal.[15]
Moving away from closed networks and what administrators can do within theirs we have two main deterrence strategies for cyber warfare with three underlying basic requirements which are capability, communication, and credibility [15]:

- Denial: physically prevent an enemy from obtaining a certain technology
- Punishment: last resort strategy in case denial has failed; prevent aggression from an enemy by threatening with greater aggression

Overall, cyber defense is the most important part of cyber warfare. Preparation and foresight are key factors for a more secure system.

5.3 Analyzing the Stuxnet Worm[11][16][17]

To put the learnings of this paper into perspective, we're going to have a look at the Stuxnet worm which is assumed to have been aimed at harming or slowing down the iranian nuclear porgram (uranium enrichment).
Stuxnet is categorized as an Advanced Persistent Threat: It was programmed very carefully and sophisticatedly with the intent of invading a very specific target system, staying undetected for the maximum amount of time, gaining more influence withing the target network while doing harm to it gradually and stealthily. Stuxnet went beyond only stealing information. Instead, it tried to manipulate certain microchips that were responsible for controlling the rotation speed of specific engines of enrichment centrifuges within the plant.
With regards to the in Chapter 2.2 mentioned phases of APT, the following can be said for Stuxnet:

- **Phase 1, Reconnaissance & Weaponization:** There's not much known about how the programmers of Stuxnet got their information, but they most definitely had insider information in various key positions. the level of insight they had to know to get to that level Stuxnet had, is imense. They knew what specific types of engines were built into the machinery as well as what software was used to control those.
- **Phase 2, Delivery:** Stuxnet made use of various zero-day exploits: It impersonated legitimate software by using stolen certificates, it installed itself on the desktop without any notifications via a compromised USB stick once it was plugged into a PC (using a flaw in the Microsoft Windows Operating System to go unnoticed), and, then, seeked out a certain version of the software *Step7* by Siemens and hacked it by applying a secret, built-in password.
- **Phase 3, Initial Intrusion to Phase 5, Lateral Movement:** After a successful installation on one target system, the worm infiltrated the network, spreading out to various other PCs, searching every one of them for the above mentioned specific version of *Step7*. After finding it, it continued on to look for a control equipment called PLC (Programmable Logic Controller) which is responsible for communication between the machinery and the PC. Once it had infiltrated the PLC, it started to check for specific types of microchips. If either of these steps proved to be unsuccessfull Stuxnet stopped its processes on that system and deinstalled itself. A high level of stealth was achieved by the hidden install exploit as well as the use of the seemingly valid, stolen certificate, making the worm appear like a legitimate software. Also, since the program altered the speed in which the centrifuges were spinning and such a modification would have attracted attention, it altered the sensor data, making it look like all processes were running like normal.

Since the goal of Stuxnet wasn't to exfiltrate data, Phase 6 will not be addressed here.
Given the attack strategies of Chapter 5.1 we can classify the Stuxnet worm as a data modification or infrastructure manipulation strategy. Its purpose was to harm machinery (centrifuges) used to enrich uranium. The motivation was most probably political, since it was feared that Iran would use the enriched material to build nuclear weapons.
Overall, Stuxnet had a great impact on the perception of

doi: 10.2313/NET-2017-09-1_10

cyber weapons since it made people painfully aware how vulnerable critical infrastructures can be to cyber attacks.

6. CONCLUSION

Cyber warfare, although seemingly more harmless than regular war, is an increasing threat. The development and growth of interconnected systems, software, and technology in general bears as many risks as it has advantage:

- Cyberwar is cheap: With a computer and Internet access, one can gain knowledge about and access to vulnerable networks.
- Due to the increasing global connectivity, malware is easy to deliver from anywhere in the world.
- Tools for attacking are free and/or open source and openly available.
- The attacker always has the advantage since he can rely on the latest updates and use the newest innovations.
- Anonymity is easily achievable so an attack might be impossible to track back to its source if the adversary is knowledgeable enough.
- Power distribution is extremely disproportional with huge gains for small actions.
- The time between the launch of an attack and its effects is barely measurable.

This list names just a few peculiarities discernible in the concept of Cyber Conflict. In addition, especially regarding open source tools, there is a threat of tools that are not publicly available. With freeware, one can download and analyze the source code, finding weak spots, counter-exploits or ways to defend against them, whereas with private tools, that is not the case.
Future wars will probably involve a mixture of conventional weaponry combined with a number of different cyber weapons. Smaller countries which would have had no chance to compete before can now enter the field, having an impact depending on their know-how and strategic approach. Nobody knows what the future will bring but one thing is for sure: If another war is coming, cyber warfare will be one of the many used with which people will harm and endanger each other.

7. REFERENCES

[1] Igor Bernik, Alan Chong, Stefania Ducci, and Joseph Fitsanakis: *Canada's Cyber Security Strategy. For a stronger and more prosperous Canada* PUBLIC SAFETY CANADA, pp. 3, `https://www.publicsafety.gc.ca/cnt/rsrcs/pblctns/cbr-scrt-strtgy/cbr-scrt-strtgy-eng.pdf`, accessed April 10th, 2017

[2] Jason Andress and Steve Winterfeld: *Cyber Warfare: Techniques, Tactics and Tools for Security Practitioners*, 2011 Elsevier, Inc, ISBN: 978-1-59749-637-7, USA

[3] John J. Arquilla and David F. Ronfeldt: *Cyberwar and Netwar: New Modes, Old Concepts, of Conflict*, Fall 1995, RAND Review, `https://www.rand.org/pubs/periodicals/rand-review/issues/RRR-fall95-cyber/cyberwar.html`, accessed April 11th, 2017

[4] Andrey Kulpin, Kal Frederick Rauscher and Valery Yaschenko: *Critical Terminology Foundations 2* East-West Institute, Russia-US Bilateral on Cybersecurity, Policy Report 2/2014, - 2014

[5] Ping Chen, Lieven Desmet, and Christophe Huygens: *A study on Advanced Persistent Threats*, iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium

[6] Ido Kilovaty: *WORLD WIDE WEB OF EXPLOITATIONS âĂŞ THE CASE OF PEACETIME CYBER ESPIONAGE OPERATIONS UNDER INTERNATIONAL LAW: TOWARDS A CONTEXTUAL APPROACH*, The Columbia Science & Technology Law Review, Vol. XVIII, STRL.ORG, Fall 2016

[7] *Gabler Wirtschaftslexikon*, Digitale Fachbibliothek, Springer Gabler, `http://wirtschaftslexikon.gabler.de/Definition/exploit.html-exploit-v3.html`, accessed April 10th, 2017

[8] Robert Axelrod and Rumen Iliev: *Timing of cyber conflict*, Ford School of Public Policy, University of Michigan, Ann Arbor, MI 48109, 2013

[9] Tom Parker, Marcus Sachs, Eric Shaw, and Ed Stroz: *Cyber Adversary Characterization: Auditing the Hacker Mind*, Syngress, 2004, USA

[10] Fyodor and David Fifield: *SecTools.Org: Top 125 Network Security Tools*, `http://sectools.org/tag/sploits/`, accessed April 11th, 2017

[11] Alexander Gamero-Garrido: *Cyber Conflicts in International Relations: Framework and Case Studies*, Engineering Systems Division, Massachusetts Institute of Technology, Explorations in Cyber International Relations, Harvard University

[12] Robin Gandhi et al.: *Dimensions of Cyber-Attacks - Social, Political, Economic, and Cultural*, IEEE Technology and Society Magazine, Spring 2011

[13] Fred Schreier: *On Cyberwarfare*, DCAF Horizon 2015 Working Paper No. 7

[14] Everett C. Dolman: *Pure Strategy: Power and Principle in the Space and Information Age*, London, Frank Cass, 2005, p. 6.

[15] Kenneth Geers: *Strategic Cyber Security*, CCD COE Publication, NATO Cooperative Cyber Defence Centre of Excellence, June 2011

[16] Ran Levi: *Stuxnet: Advanced Persistent Threat*, Curious Minds Science Technology, `http://www.cmpod.net/all-transcripts/stuxnet-the-malware-that-struck-the-iranian-nuclear-program-text/`, accessed May 7th, 2017

[17] Symantec Corporation: *Advanced Persistent Threats: A Symantec Perspective*, White Paper, 2011, `https://www.symantec.com/content/en/us/enterprise/white_advanced_persistent_threats_WP_21215957.en-us.pdf`, accessed May 7th, 2017

Malware Detection in Secured Systems

Claes Adam Wendelin
Betreuer: Stefan Liebald, M. Sc.
Seminar Future Internet SS2017
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: wendelin@in.tum.de

ABSTRACT

This paper presents the state-of-the-art in malware detection and what properties of malware they depend on. Understanding the connection between different malware and detection methods is required for further research and design of tomorrow's malware detection. The value of information is big today considering trade secrets and the amount of personal data available on our computer systems.

Three analysis methods are covered. Static analysis detects simple malware well, but has known flaws that can be used to circumvent detection. Dynamic analysis with its focus on recognizing behavior has robustness and generalization, but raises questions regarding the future complexity of the analysis. Automating malware detection with machine learning shows potential although the current techniques suffer from a high false positive rate.

The kernel is a source of trust in malware detection, which testaments the importance of its integrity. Preventing malware from hijacking the kernel can be done with signed drivers or protection of function pointers.

Keywords

Malware Detection, Static Analysis, Dynamic Analysis, Heuristic Analysis, Kernel Integrity

1. INTRODUCTION

Malicious software (malware) is software designed to cause unwanted behavior on a computer. Incentives to write and deploy malware can be economic, cultural, social, or politic [20], which can affect vital parts of society such as healthcare, finance and trust. The United Kingdom government estimated in 2011 that the overall cost of cyber crime to its economy amounts to £27 billion [12].

A malware infected computer can be used as a resource to spread the malware further or provide services such as spam. A specific network of infected computers, Cutwail, was reported by Symantec [34] to be responsible for 46.5% of spam on the Internet and estimated by [43] to have made profits from $1.7 million to $4.2 million in the time period 2009-2011.

Malware can also be used to gain unauthorized access to information. Trade secrets are an important economic asset to companies. Hiding information instead of making it generally known can give an advantage to businesses, but a trade secret has no exclusive rights compared to a patent, which makes them a valuable target for cyber espionage. [38]

This paper reviews malware detection techniques and what aspects of malware they depend on. Background to the requirement of sophisticated detection methods is covered in Section 2. Two system penetration techniques are listed in Section 3 and followed by detection methods in Section 4. Various ways to protect kernel integrity is found in Section 5. Finally the methods are assessed and concluded in Section 6 and 7 respectively.

2. BACKGROUND

Advanced Persistent Threats (APT) are malware designed to target a specific organization with often custom written malware not yet recognized by anti-malware detection [7]. Thomson [47] addresses the risk of APTs and the difficulty of detecting them due to their specialized nature. An APT has long term goals and propagates slowly and carefully to get to its objective, which is greatly contrasted by most traditional malware that operates on a hit-and-run basis [8]. Anti-malware software has been developed to focus on the common malware that tries to spread and act as fast as possible to infect the widest possible target group before being analyzed and protected against. The meticulous nature of APT in addition to its narrow target group reduces the chance of it being discovered and analyzed by commodity anti-malware. More advanced detection methods are therefore needed to combat the APTs.

An important part of APTs lies in the preparation for the attack. The target group and system must be studied closely to enable system penetration and avoiding detection. System penetration methods are discussed further in Section 3 and is the first real contact between the target and the malware. The exploit can be launched using a small infection file, that after compromising the system and stealing valid system credentials, downloads the real malware modules and then erases itself to hide the intrusion [7]. The APT is then on the system with valid user access and no direct connection with the original infection point.

The custom written nature of the APT means that anti-malware has no record of it and therefore no immediate reason to suspect its files. Analyzing the behavior is also a challenge as it attempts to mirror user behavior to prevent its valid access from getting revoked. While covert on the system, the APT tries to get access to the desired data,

doi: 10.2313/NET-2017-09-1_11

which is to be exported outside the system for collection.

3. SYSTEM PENETRATION

The first step of malware propagation is to infect a host with the malware. There exists a wide variety of methods to distribute malware, but the two that will be covered here are social engineering and quantum insert. Both have in common that they can be used for malware attacks against specific targets, a desirable trait for APTs.

3.1 Social Engineering

Social engineering is a mix of psychology, human nature and manipulation. It is used to manipulate people to do something desirable for you even though it might be undesirable for them. Social engineering is comparable to a magician, which with subtle cues masterfully diverts your attention to whatever they want while they perform the magic.

Technical countermeasures to malware are useless if a user can be made to bypass them. People are often called the weakest link in the security system and it is their naivety and trust that social engineering methods try to exploit. A simple trust in a friend sending you a link via social media can be enough to compromise a system. The botnet Koobface [46] has reportedly access to over 1800 compromised hosts, access which was gained by spamming enticing links on Twitter and Facebook. Infected accounts become part of the spreading as they are used by the botnet to further reach out to their friends with links.

Emotions are an important part of social engineering. Instilling feelings such as fear, greed, sympathy or anger can make us do things we otherwise would not do and have been abused since the beginning of time. Abraham et al. [1] summarizes social engineering malware principles and draws the conclusion that to combat social engineering - purely technical solutions are not enough. They suggest improvements by raising awareness, further monitoring, security policies and motivating users to follow secure practices.

Phishing is a social engineering technique where the user is tricked into giving away sensitive information, often through e-mails or social media, by disguising itself as a legitimate source [45]. According to the Anti-Phishing Working Group, the number of phishing attempts increased by 65% from 2015 to 2016 yielding a total of 1,220,523 attempts in 2016 [3].

3.2 Quantum Insert

Quantum insert is a technical approach to implant malware on a targeted set of users by utilizing an inherent flaw in TCP. *The Intercept* has reported usage of Quantum Insert by the governmental intelligence bureaus NSA [19] and GCHQ [18] to facilitate unauthorized surveillance and monitoring of traffic.

When a desired target opens up a connection with a website, the attacker snoops on the TCP packet containing the HTTP request and quickly spoofs an answer packet that would redirect the target to an infected website instead [22]. All required information to create a valid answer packet exists in the outgoing packet, so the target's system will accept the first one that reaches it. This puts requirements on the attack, since the attacker's server must be in close vicinity to the target to get a minimal latency.

Performing an attack like this requires processing of huge amounts of data in real-time to find interesting users and a nearby server infrastructure that can send the spoofed packet and serve a malicious website. An infection similar to Quantum Insert is China's Great Cannon [29], which intercepts and injects malicious Javascript code into packets with a set of target addresses.

Detecting Quantum Insert is not straightforward, but [22] suggests multiple symptoms, which can indicate an ongoing attack: (1) duplicate TCP packets with different payloads, (2) anomalies in Time-To-Live values due to different hop counts in the routes and (3) other inconsistent values in the TCP header. Although duplicate TCP packets could indicate Quantum Insert, being too considerate with duplicates might make the system significantly weaker to denial-of-service attacks using flooding [33].

4. DETECTION METHODS

The following sections will reason about detection methods used by anti-malware and various countermeasures by the malware to avoid them. An aspect of the malware that can be used for identifying it as such is called a feature. The detection methods focus on different parts of the malware to extract features from.

There are two ways in general to analyze malware. Dynamic analysis, which is described in Section 4.2, and static analysis described below.

4.1 Static Analysis

Static analysis is done without running any of the malware's code. The malware therefore will not be executed by the computer unless deemed safe by the analysis. A common technique used in static analysis is syntactic signature detection, but alternative detection methods with semantic signatures and heuristics have been proposed.

A first step before the malware code can be deconstructed and analyzed, is unpacking and/or decryption [35]. Malware uses compression to reduce the size of their executable, which causes a lower impact transfer of the malware to the to be infected system. Unpacking a file requires knowledge of the packing method, which means that anti-malware software can not always unpack them. Denying packed files altogether is not a feasible solution, since legitimate software also utilizes packing.

The simplest version of malware utilizing packing would consist of a malign binary, which is the payload, and decryption code that can unpack the malign code and run it. Since in such malware, the decryption code is constant, anti-malware can analyze the decryptor and recognize malware if it has been detected before [57]. Another option to combat packing is to let the malware unpack itself and analyze its code in memory using dynamic analysis, which is covered in Section 4.2.

4.1.1 Syntactic Signature Detection

doi: 10.2313/NET-2017-09-1_11

This technique utilizes that malicious instructions or code sequences of a known malware can be used to detect the same malware again. Mapping an arbitrary length of data to a fixed size can be done using a hash function. Important aspects of the hash function is that it is deterministic to always produce the same output for the same input, and that it evenly distributes the input data over the output to reduce the risk of collisions. By calculating a hash over the malign code, you get a syntactic signature. Anti-malware software collects known malware's syntactic signatures in a database, which allows them to be detected in the future. A syntactic signature detector is dependent on the database to detect malware, which makes it prone to new or changed malware.

Malware can then be detected by using regular expressions or pattern matching to compare a file's content to the signature database. A matching signature means that malicious code has been spotted in the scanned file. Finding malicious behavior like this is highly dependent on a consistent syntax of the code, since different, misplaced or additional instructions will change the hash output. Robustness against syntax changes can be improved by analyzing the file's content in chunks to generate a series of signatures [41]. A percentage of matching signatures can be enough to recognize malware.

Syntactic signature detection in static analysis has two major malware types that it has problems detecting: metamorphic malware and polymorphic malware. Their common denominator is their ability to constantly transform their syntactic signature to avoid being categorized by a syntactic signature database.

Polymorphic malware consists of an encryptor, decryptor and a constant payload that it tries to hide from the signature detector. The weakness of the earlier mentioned packing is that it is constant, so once it is part of the signature database, it can be detected again. Polymorphic malware gets rid of that weakness by mutating its static payload and decryptor. For every propagation, the polymorphic malware generates a new encryption key, which the encryptor uses to pack the static binary. Using a different key every encryption together with a different decryptor ensures that all copies of the malware have a different syntactic signature. [15]

The decryptor is generated by a polymorphic engine using various code obfuscation techniques to make it harder to analyze and different from previous versions. Different techniques include[9, 57]:

- Nop-insertion, the addition of no operation instructions.
- Code transposition, switching the order of independent instructions.
- Register reassignment, changing the used registers for instructions.
- Code substitution, exchanging instructions with semantically equivalent ones.

Creating an polymorphic engine that consistently produces distinct copies without being recognizable is usually more sophisticated than the malware itself, which can manifest in flaws that aid in its detection[15, 26].

Polymorphic malware does not mutate its behavior and instead hides it with encryption. A malware type heavily involved in obfuscation rather than encryption is the metamorphic malware, which does not need encryption to stay hidden. Instead, it fundamentally changes its code every propagation to always have different syntactic signatures [15]. The metamorphic malware is transformed using the same code obfuscation techniques as the polymorphic malware, but the scope is increased to the whole malware.

The process of metamorphic malware's mutation can include up to five steps [37, 35].

Firstly, the malware must decode its instructions into an intermediate form that describes its functionality. Combinations of different instructions can be functionally equivalent when combining certain arguments, which makes the extraction a complex task.

Secondly, the metamorphic engine shrinks the intermediate form by removing instructions that does not do anything and was included by the previous mutation. Without this step, metamorphic malware could gain size every mutation and therefore increasing its workload every iteration. Keeping the final code a similar size is important to avoid giving anti-malware features that can be tracked.

Thirdly, the intermediate form's control flow is changed by reordering instructions and submodules, which are linked together with jump instructions. The metamorphic engine must analyze which instructions are independent from each other to retain the functionality while transforming as much as possible.

Fourthly, no-operation instructions and other functionally irrelevant instructions are added to expand the malware once more. This addition makes sure that syntactic signature detection of code sections with low permutability is improbable.

Finally, the metamorphic engine reassembles the malware's intermediate form back into instructions for the infected system.

The metamorphic engine tries to make the code as hard as possible to analyze, which further makes the transformation of itself more complicated. Because of an increased amount of instructions from the imperfect obfuscation, the CPU workload is also increased. Anti malware software can monitor CPU idle time and set benchmarks, which can lead to detection of a malware's workload signature [35].

4.1.2 Semantic Signature Detection

Malware detection methods focused on analyzing the syntactic nature of the malware have weaknesses toward code obfuscation. Semantic signature detection avoids those flaws by focusing on the behavior of the program. Variants of the malware, that would possibly require different syntactic sig-

doi: 10.2313/NET-2017-09-1_11

natures, all have a similar functional flow [24]. Multiple variants of the same malware can therefore be detected using just one semantic signature.

Semantic signature detection is being explored as a valid alternative to syntactic signature detection for static detection. Christodorescu et al. [11] introduces formal semantics to classify malicious behavior.

Formal semantics acts as a specification language with which you can define behavior using a template. Templates consist of a sequence of instructions, variables and symbolic constants. Specific registers and values in the malware's code are abstracted away in the template, but their dependencies are preserved. The templates are therefore not affected by common code obfuscation techniques such as register reassignment and no-operation instructions. Malicious behavior is specified in templates and compared to the dubious binary to determine if it is malicious or not.

A similar semantic approach has been explored in [24], where Kinder et al. uses model checking to verify malware specifications. Model checking is a way to verify the correctness of a finite-state system against a specification (e.g. only one process allowed in a critical section). In this case, the system is verified against unwanted behavior, which is defined as a set of states, transitions and labels. The malware binary is disassembled into machine instructions that are used to build a model of the program, which is verified with model checking against specified malware behavior.

Both of these semantic analyses try to construct a control flow of the target binary to determine whether it is malign or benign. By making the static analysis harder using binary obfuscation, malware could avoid detection. Moser et al. [32] point out that a vital part of semantic analysis lies in establishing relationships between constants. Constants in the binary could be used for arithmetic operations or as jump destinations. Obfuscating jump destinations can make control flow analyses miss the malicious set of operations that would otherwise be uncovered. Obfuscating binary constants using the known hard to solve 3-satisfiability problem can make the derivation of the constants in static analysis an NP-hard problem [32].

4.2 Dynamic Analysis

In contrast to static analysis, dynamic analysis actually executes the malware and from that tries to deduce if its malign or not. There are primarily two ways to dynamically analyze malware that are used by anti-malware software. Either of them uses emulation to execute the malware in a safe environment to hopefully reveal its intent or monitor a program's behavior in real time to detect and prevent suspicious actions before executed [2]. The most common analysis method used for dynamic analysis is with behavioral traits on the basis that malware is classified by what it does instead of what it looks like. As system calls are the interface to the operating system, monitoring those can reveal a program's resources and actions [16]. Lots of papers can be found that utilizes system call monitoring in a dynamic analysis fashion, including but not limited to [21, 30, 10, 36, 25, 27]. Network traffic can also be used to characterize malware [21].

Dynamic analysis is inherently resistant to the code obfuscations mentioned in Section 4.1 that tries to hinder static analysis. As the actually executed instructions are considered in dynamic analysis, obfuscated code can be identified after its decryption to valid instructions. For example, obfuscated constants [32] that hide jump addresses or data dependencies behind complex calculations will unwind themselves into proper instructions when executed, which makes them easy to analyze with dynamic analysis. Polymorphic malware is also easier to analyze with dynamic methods, since its decryptor will unpack its static payload into memory to let it run. Static payload is easily identified by syntactic signature detection.

A flaw of dynamic analysis is that only one execution path of the monitored program is revealed. Malware can exploit this by detecting that they are being monitored and hide malicious functionality or immediately exit. To do this, malware can either look for clues of specific analysis tools in files, registry or processes, or by analyzing imperfections that occur during emulation such as timing properties or variations of instructions [5].

Malware can not be analyzed forever, which means that *logic bombs*, malware that triggers on an event, can go undetected by dynamic analyzers. Logic bombs could trigger on a date, a user input or remote control via network [14]. The event that triggers the malware is often external, so tracking everything that the malware reads together with its control flow decisions can be used to simulate different inputs to the malware and reveal its multiple execution paths [31].

Ether [13] is a malware analyzer that uses hardware virtualization extensions to allow it to reside entirely outside the target operating system environment. Thereby no trace of the analyzer can be found in the emulation. Another analyzer trying to stay transparent to malware is Cobra [48]. Cobra disassembles and executes malware code in blocks that are scanned for instructions that might give away the environment. Such instructions are replaced with safe versions that will not give away the analysis tool. Both of these tools incur a significant performance overhead, which can be undesirable. Another technique developed for analysis-aware malware introduced in [5] compares the execution of malware in an analysis environment with execution in a reference environment. If the execution differs, the malware should be analyzed more thorough.

4.2.1 Semantic Signature Detection

Semantic signature detection tries classify malware by creating malicious behavior signatures. The state of the art in semantic signature detection is using a behavior graph, with which malware detection is an NP-complete problem [17]. Behavior graphs are composed out of nodes, which represent actions (e.g. system calls), and edges that represent data dependencies between nodes. Behavior graphs can be used as a detection method by monitoring an unknown program's system calls with their dependencies, and match that behavior with nodes in the graph. An enough matched graph indicates malicious behavior [25].

On an end-system, the semantic signature would be detected after the malicious activity has been performed, which puts

a clock on the anti-malware to prevent the malware from doing further damage and repair what was done. A too slow detection system might give the malware enough time to complete its objective. The complexity of behavior graphs has lead to the development of a lightweight alternative to behavioral graphs. Lu et al. [27] suggest a solution that generalizes system calls by classifying them based on the resource used (e.g. file, socket) and a classification of those resources based on similar functionality. The abstraction level is higher than a behavior graph, which makes it robust against code-obfuscation.

The current state of the art behavior graphs focus their analysis on a single process to reduce the complexity and computation power required. Ma et al. have shown [28] that multi process malware can automatically be generated from a malware's source code and successfully avoid semantic detection in single process based malware detector. The technique to generate multi process malware is shown in figure 1. The process P_0's two system calls are split up between two processes P_0' and P_1'. The state change caused by the system call is transferred between the processes so that the execution of P_0 is equivalent to the execution of P_0' and P_1'.

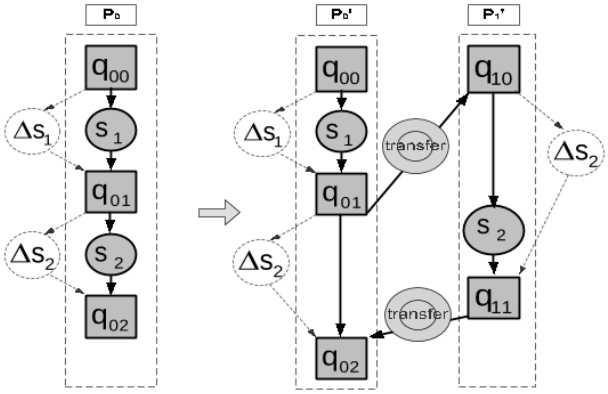


Figure 1: Creation of multi process malware from single process malware. Source: [28]

4.3 Heuristic Analysis

Heuristic methods make use of data mining and machine learning techniques to try and detect malware. Such techniques try to classify malware using knowledge from previous data that can either be labeled or not. The algorithm uses a feature vector as an input and outputs a classification of the supposed to be malware that is either malign or benign. A feature vector contains different information about the malware that is used in the classification process. The chosen features are vital for the success. Five features are discussed in [6]:

- API/System calls: a program's requests to the operating system.
- Control flow graphs: a program's statements and control flow to construct a directed graph.
- N-Grams: substrings of a program's binary code.
- OpCode: sequences and statistical frequencies of different machine language instructions.
- Hybrid features: a combination of feature types.

The feature vector is built from the chosen data. Classification based on system calls could use sequences of system calls as input, which the algorithm processes to determine if its an illicit sequence.

Yan et al. [55] have made a study of useful features in automated malware classification. Their results point out that the header of a Windows executable is highly discriminatory. But there is a dilemma, since an adoption of the technique might lead to feature obfuscation by malware writers. Robustness of features is therefore an important issue and further discussed in [56].

Machine learning techniques can be used to detect metamorphic malware. A detection technique using Hidden Markov Models is proposed in [54], where a model is trained on a family of metamorphic malware to be able to recognize its defining features. The model becomes specialized in one malware family, but managed to also detect some malware from another family. The same weakness of similarity is exploited by Runwal et al. [40] in a detection technique involving the similarity score between the opcode graphs of metamorphic families and normal files.

Heuristic analysis can be a very powerful technique considering its automation possibilities, but as [6] summarizes: heuristic malware detection suffers from a high false positive ratio.

5. PROTECTING KERNEL INTEGRITY

The kernel of an operating system has complete control of the system. It handles start-up, input and output from programs, peripherals, memory and CPU. Access to the kernel can be used to extend the operating system with malign functionality such as backdoors, logging keystrokes, escalating privilege of malware or tampering with other defense mechanisms. The premise of malware detection methods is that they have higher privilege than the malware, which is invalid if the malware has kernel access.

The malware type rootkit will try to stay covert after infection while it fulfills its illicit objective. A typical hiding method is to remove its entry from the system processes list. Kernel rootkits are malware that targets the kernel and actively tries to tamper with it.

Baliga et al. [4] propose a general classifying scheme for rootkit attacks, which are categorized by the tampering technique: *Control hijacking* and *control interception* both manipulate the control flow of the kernel by changing the system call table, interrupt descriptor table and kernel code. *Control tapping* will ensure that the attack code is executed on every invocation of a specific function without affecting it. *Data value manipulation* tries to indirectly manipulate the kernel by changing values of critical variables. *Inconsistent data structures* similarly tamper with kernel data structures, which are assumed to be consistent, but are made inconsistent and can be used to hide processes and modules. A rootkit can combine a variation of these to accomplish its task on the system.

doi: 10.2313/NET-2017-09-1_11

Defense against rootkits can be divided into prevention and detection. The goal of prevention is to avoid unauthorized access to the kernel by keeping its integrity consistent. A method of doing so is kernel module signing. Such a policy requires all kernel modules to have an embedded digital signature, which can be verified against a source of trust. Windows requires signed drivers by default [52] - a feature that only can be disabled via a reboot. As [23] mentions, kernel module signing security is made on the assumption that all loaded code in the kernel is safe and prevents execution of unsigned kernel code. The Turla (a.k.a. Uroburous, Snake) [44] rootkit managed to bypass Windows' driver signature policy by leveraging a flaw in the virtualization tool VirtualBox [49], which included a signed driver containing a privilege escalation vulnerability.

Detection on the other hand tries to safeguard the kernel integrity by detecting unexpected changes in critical regions. As function call hooking is often a vital part of rootkits, detecting and preventing the hijacking could protect the kernel's integrity. Detecting illegitimate function pointers (hooks) requires a reference to compare with. Some detectors [50, 58, 51, 39] dynamically find kernel hooks, which they use shadow memory to keep consistent. Shadow memory is a technique that maps memory bytes to metadata. The metadata could keep track of the number of writes to a specific memory section or also if the memory originates from a trusted source.

Verifying that only trusted sources modify function pointers could protect their integrity and prevent rootkits. Hardware-based memory protection can safeguard the function pointers, but it only protects on a page-level granularity. Data that does not need memory protection, but is on the same page as a function pointer, would get an access overhead using such a technique.

A novel approach by [50] is to generate a shadow copy of all function pointers in a centralized page-aligned memory location depicted in figure 2, where hardware-based protection is utilized efficiently. Function pointers are accessed through an indirection layer with either read or write operations. A read request will be fulfilled directly by the indirection layer and a write operation will be passed on to the hardware-based memory protection to validate the request.

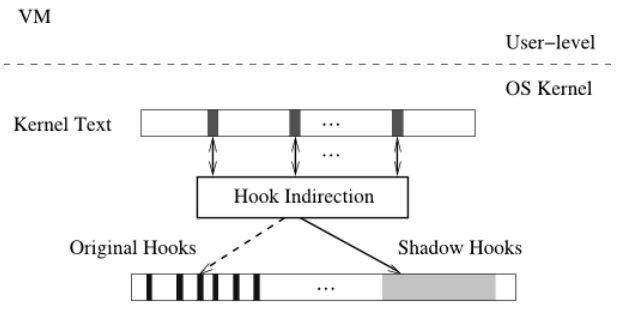


Figure 2: Redirection of function pointers to protect their integrity. Source[50]

Windows has an approach similar to shadow memory that saves data about kernel function pointers and interrupt tables to allow validation. The data can be known-good copies and metadata in the form of checksums, which allows the operating system to validate kernel integrity during run-time to prevent tampering [53, 42].

Although preserving kernel integrity thwarts the majority of rootkits, such a technique can be bypassed using return-oriented rootkits [23]. Return-oriented rootkits construct malicious stack frames by using *gadgets*. A *gadget* is a combination of kernel instructions that together form a well-defined behavior, e.g. computing AND of two operands. The self-contained *gadgets* can be combined to manipulate the stack and launch attacks upon the system. Integrity checks and memory protection fails since all malicious activity is done with safe instructions already in the kernel.

6. ASSESSMENT

Static analysis and syntactic signature detection methods are a great first defense against malware. Using a dictionary of previously known malware makes it harder for old malware to infect systems again, but it is inherently weak against new and advanced malware.

Semantic detection in static analysis is powerful due to its complete analysis of all the malware's execution paths and the generality of its signatures. Future malware using an already known way to achieve its objective can be detected without updating the signature database. The limited amount of system calls suggests that a thorough cataloging of malicious sequences are possible.

An issue with semantic detection arises with the increased complexity of analysis that obfuscation can yield. Most of the semantic techniques try to construct a behavior signatures from graphs, but correlation between system calls and actions are not always obvious. A malware split up into multiple processes requires a more advanced analysis to try find data or control dependencies between processes. The analysis becomes even harder if dependencies between processes are obfuscated by using a non-conventional messaging system. Process A and B could communicate with different servers, which relay messages between each other and thereby hide the connection between process A and B.

Dynamic analysis is unfazed by encryption methods and obfuscated constants, which testaments its relevancy besides static analysis. As of now, dynamic analysis can be efficient against traditional malware that acts on a rapid pace, but is lacking if the malware is patient. An APT that can hide and stay inactive for months is hard to detect using dynamic analysis as the malware does not generate any evidence against itself until it actually acts.

Protecting against advanced malware and especially new malware requires a general and automatic classification system that can perform without human intervention. Heuristic methods are not mature enough yet, but their success in other areas showcase their power in classification based on features, which makes it a promising research area. Another important area is kernel integrity, since a breached kernel can tamper with anti-malware and do substantial damage to the system.

doi: 10.2313/NET-2017-09-1_11

7. CONCLUSION

There exists a wide variety of malware detection methods, but none of them are robust to all malware. Different advantages and disadvantages in analysis methods suggests that a detector should not be limited to one method, but rather utilize a wide set of different tools to complement each other.

The possibilities to automatically increase the complexity of static analysis and dynamic analysis highlights the challenges that anti-malware software faces. Automatically detecting new malware is an important topic, where machine learning is a promising area. Machine learning has made breakthroughs the last years that will hopefully spread to malware detection.

8. REFERENCES

[1] S. Abraham and I. Chengalur-Smith. An overview of social engineering malware: Trends, tactics, and implications. *Technology in Society*, 32(3):183–196, 2010.

[2] E. Al Daoud, I. H. Jebril, and B. Zaqaibeh. Computer virus strategies and detection methods. *Int. J. Open Problems Compt. Math*, 1(2):12–20, 2008.

[3] APWG. Phishing Activity Trends Report, 4th Quarter 2016. `https://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf`. Accessed: 2017-04-09.

[4] A. Baliga, P. Kamat, and L. Iftode. Lurking in the shadows: Identifying systemic threats to kernel data. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 246–251. IEEE, 2007.

[5] D. Balzarotti, M. Cova, C. Karlberger, E. Kirda, C. Kruegel, and G. Vigna. Efficient detection of split personalities in malware. In *NDSS*, 2010.

[6] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh. A survey on heuristic malware detection techniques. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 113–120. IEEE, 2013.

[7] R. Brewer. Advanced persistent threats: minimising the damage. *Network security*, 2014(4):5–9, 2014.

[8] P. Chen, L. Desmet, and C. Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[9] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. Technical report, DTIC Document, 2006.

[10] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *Proceedings of the 1st India software engineering conference*, pages 5–14. ACM, 2008.

[11] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant. Semantics-aware malware detection. In *Security and Privacy, 2005 IEEE Symposium on*, pages 32–46. IEEE, 2005.

[12] Detica and C. Office. The cost of cybercrime: A detica resport in partnership with the office of cyber security and information assurance in the cabinet office. Technical report, Cabinet Office and National security and intelligence, 2011.

[13] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62. ACM, 2008.

[14] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2):6, 2012.

[15] P. Ferrie and P. Ször. Hunting for metamorphic. *Virus, págs*, pages 123–143, 2001.

[16] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 120–128. IEEE, 1996.

[17] M. Fredrikson, M. Christodorescu, and S. Jha. Dynamic behavior matching: A complexity analysis and new approximation algorithms. In *International Conference on Automated Deduction*, pages 252–267. Springer, 2011.

[18] R. Gallagher. Operation socialist: The inside story of how british spies hacked belgium's largest telco. *The Intercept*, 12, 2014.

[19] R. Gallagher and G. Greenwald. How the nsa plans to infect 'millions' of computers with malware. *The Intercept*, 12, 2014.

[20] R. Gandhi, A. Sharma, W. Mahoney, W. Sousan, Q. Zhu, and P. Laplante. Dimensions of cyber-attacks: Cultural, social, economic, and political. *IEEE Technology and Society Magazine*, 30(1):28–38, 2011.

[21] A. R. Grégio, D. S. Fernandes Filho, V. M. Afonso, R. D. Santos, M. Jino, and P. L. de Geus. Behavioral analysis of malicious code through network traffic and system call monitoring. In *SPIE Defense, Security, and Sensing*, pages 80590O–80590O. International Society for Optics and Photonics, 2011.

[22] L. Haagsma. Deep dive into quantum insert. *Online at https://blog.fox-it.com/2015/04/20/deep-dive-into-quantum-insert*, 2015.

[23] R. Hund, T. Holz, and F. C. Freiling. Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms. In *USENIX Security Symposium*, pages 383–398, 2009.

[24] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith. Proactive detection of computer worms using model checking. *IEEE Transactions on Dependable and Secure Computing*, 7(4):424–438, 2010.

[25] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-y. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *USENIX security symposium*, pages 351–366, 2009.

[26] E. Konstantinou and S. Wolthusen. Metamorphic virus: Analysis and detection. *Royal Holloway University of London*, 15, 2008.

[27] H. Lu, B. Zhao, J. Su, and P. Xie. Generating lightweight behavioral signature for malware detection in people-centric sensing. *Wireless personal communications*, 75(3):1591–1609, 2014.

[28] W. Ma, P. Duan, S. Liu, G. Gu, and J.-C. Liu. Shadow attacks: automatically evading system-call-behavior based malware detection. *Journal in Computer Virology*, 8(1):1–13, 2012.

[29] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson. China's great cannon. *Citizen Lab*, 10, 2015.

[30] L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. C. Mitchell. A layered architecture for detecting malicious behaviors. In *International Workshop on Recent Advances in Intrusion Detection*, pages 78–97. Springer, 2008.

[31] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 231–245. IEEE, 2007.

[32] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual*, pages 421–430. IEEE, 2007.

[33] R. M. Needham. Denial of service. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 151–153. ACM, 1993.

[34] Nisbet. Cutwail Takedown Cripples Bredolab Trojan; No Effect on Spam Levels. `https://www.symantec.com/connect/blogs/cutwail-takedown-cripples-bredolab-trojan-no-effect-spam-levels`. Accessed: 2017-04-09.

[35] P. O'Kane, S. Sezer, and K. McLaughlin. Obfuscation: The hidden malware. *IEEE Security & Privacy*, 9(5):41–47, 2011.

[36] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel. Fast malware classification by automated behavioral graph matching. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 45. ACM, 2010.

[37] F. Perriot, P. Ször, and P. Ferrie. Striking similarites: Win32/simile and metamorphic virus code. *Symantec Corporation*, 2003.

[38] J. Pooley and D. P. Westman. Trade secrets. In *WIPO Magazine*. Law Journal Seminars-Press, 1997.

[39] R. Riley, X. Jiang, and D. Xu. Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. In *International Workshop on Recent Advances in Intrusion Detection*, pages 1–20. Springer, 2008.

[40] N. Runwal, R. M. Low, and M. Stamp. Opcode graph similarity and metamorphic detection. *Journal in Computer Virology*, 8(1-2):37–52, 2012.

[41] W. Scheirer and M. C. Chuah. Syntax vs. semantics: competing approaches to dynamic network intrusion detection. *International Journal of Security and Networks*, 3(1):24–35, 2008.

[42] Skywing. Patchguard reloaded: A brief analysis of patchguard version 3. *Uninformed Journal*, 8(5), Sept. 2007.

[43] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. *LEET*, 11:4–4, 2011.

[44] B. Systems. The Snake Campaign: Cyber Espionage Toolkit. `http://www.baesystems.com/en/cybersecurity/feature/the-snake-campaign`. Accessed: 2017-04-09 Last updated: 2016-01.

[45] C. Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[46] K. Thomas and D. M. Nicol. The koobface botnet and the rise of social malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 63–70. IEEE, 2010.

[47] G. Thomson. APTs: a poorly understood challenge. *Network Security*, 2011(11):9–11, 2011.

[48] A. Vasudevan and R. Yerraballi. Cobra: Fine-grained malware analysis using stealth localized-executions. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

[49] VirtualBox. `https://www.virtualbox.org/`.

[50] Z. Wang, X. Jiang, W. Cui, and P. Ning. Countering kernel rootkits with lightweight hook protection. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 545–554. ACM, 2009.

[51] Z. Wang, X. Jiang, W. Cui, and X. Wang. Countering persistent kernel rootkits through systematic hook discovery. In *International Workshop on Recent Advances in Intrusion Detection*, pages 21–38. Springer, 2008.

[52] Windows. Driver Signing Policy. `https://msdn.microsoft.com/en-us/windows/hardware/drivers/install/kernel-mode-code-signing-policy--windows-vista-and-later-`. Accessed: 2017-04-09 Last updated: 2016-09-08.

[53] Windows. Kernel Patch Protection. `https://msdn.microsoft.com/en-us/library/windows/hardware/Dn613955(v=vs.85).aspx`. Accessed: 2017-04-09 Last updated: 2007-01-22.

[54] W. Wong and M. Stamp. Hunting for metamorphic engines. *Journal in Computer Virology*, 2(3):211–229, 2006.

[55] G. Yan, N. Brown, and D. Kong. Exploring discriminatory features for automated malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 41–61. Springer, 2013.

[56] C. Yang, R. C. Harkreader, and G. Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *International Workshop on Recent Advances in Intrusion Detection*, pages 318–337. Springer, 2011.

[57] I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 297–300. IEEE, 2010.

[58] A. Zaki and B. Humphrey. Unveiling the kernel: Rootkit discovery using selective automated kernel memory differencing. In *Proceedings of the 2014 VIRUS BULLETIN CONFERENCE*, 2014.

doi: 10.2313/NET-2017-09-1_11

ISBN 978-3-937201-58-0
DOI 10.2313/NET-2017-09-1
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)