technische universität
dortmund

Master Thesis

Recognition of Attackers in the Context of Honeypots

Patrick Trockel
August, 2018

Advisors:
Prof. Dr. Peter Martini
Prof. Dr. Jian-Jia Chen

TU Dortmund University
Faculty of Computer Science XII
Embedded Systems (LS12)
https://ls12-www.cs.tu-dortmund.de

In Cooperation with:
Bonn University
Fraunhofer FKIE

Abstract

According to the definition of a honeypot, any connection to or from this resource can be considered as malicious and therefore as an attack. After detection, in particular, the recognition of attackers or targeted attacks is a serious problem: Several sessions of possibly multiple attackers must be linked reliably.

For this purpose, an approach for targeted attacks is presented in this thesis which uses a transparent proxy and a high-interaction honeypot as a foundation. These are integrated into a complex architecture and allow an attacker to reach an isolated network segment - the HoneyLab. Apart from the architecture active and passive measures are used to recognize the attacker. Some of these include injection of unique credentials and URLs, client and browser fingerprinting and a comprehensive data recording.
The main concept is to let the attacker build up a unique and compromised knowledge. If the attacker uses this knowledge in several attacks or sessions, it will be used against him which leads to recognition.

The implemented approach is evaluated with security professionals and random real-world attackers. The security professionals were asked to behave as if they were participating in a Capture the Flag (CTF) event and were reliably recognized by the approach. The random real-world attackers could be detected but not recognized because they only used automatic scans without any human interaction or malicious activities. Since targeted attacks and not automatic scanners were focused, this can nevertheless be considered a success.

Acknowledgements

First, I would like to thank my two examiners Prof. Dr. Peter Martini from the Bonn University and Prof. Dr. Jian-Jia Chen from the TU Dortmund University, who made it possible for me to work on this thesis within cooperation. Also, I would like to thank Dr. Elmar Padilla, Simon Ofner and Tim Krause from the Fraunhofer FKIE (Department of Cyber Analysis & Defense) who have made the topic available to me and have always supported me.

A special thank goes to my supervisor Tim Krause: Thank you very much for your time during periodic phone calls and appointments and the excellent supervision. Through the qualified suggestions and tips, I always felt very comfortable and in good hands.

Many thanks also to Lars Dröge and Gerd Sokolies from the TU Dortmund University who have set up and provided me with the resources in the form of hardware and the separated network segment. Thank you for the technical support.

Furthermore, I want to thank my *known attackers* who have spent a lot of time to test and attack my research setup. Thank you very much, Denis Berger, Daniel Forster, Roland Kühn and Fabian Mosch.

And last but not least, I want to give a special thanks to my girlfriend, Ann-Kristin Kaltefleiter-Jürgens. Thank you for your unceasing advice and support during this thesis and thank you for all the compromises you've made during this time.

Contents

List of Abbreviations

3DES	Triple Data Encryption Standard
API	Application Programming Interface
APT	Advanced Persistent Threat
CBC	Cipher Block Chaining
CMS	Content Management System
C&C-Server	Command-and-Control-Server
CERT	Computer Emergency Response Team
DDoS	Distributed-Denial-of-Service
DGA	Domain Generation Algorithms
DMZ	Demilitarized Zone
DNS	Domain Name System
HL	HoneyLab
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control System
IDS	Intrusion Detection System
IoC	Indicator of Compromise
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
MQTT	Message Queuing Telemetry Transport

NAT	Network Address Translation
NIC	Network Interface Card
OPC	Open Platform Communications
OPC UA	OPC Unified Architecture
OS	Operating System
OT	Operational Technology
PAM	Pluggable Authentication Modules
PoC	Proof of Concept
RDP	Remote Desktop Protocol
SCADA	Supervisory Control And Data Acquisition
SIEM	Security Information and Event Management
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Tactics, Techniques and Procedures
URL	Uniform Resource Locator
VM	Virtual Machine
VPN	Virtual Private Network

Chapter 1

Introduction

Information Technology (IT) Security offers various solutions in order to counter attacks on computer systems. All these measures reduce the attack surface to limit the attacker and minimize the possible attack vectors. Conventional approaches such as antivirus software, firewalls, and client and server hardening attempt to prevent attacks; other ones like network segmentation limit the scope of impact and still other, for example, Intrusion Detection Systems (IDS) and Security Information and Event Management (SIEM) tools monitor resources to detect attacks or even intrusions. Whether "next generation" or "classic" approach is not relevant, nor if it implements techniques such as behavior analysis, signature-based detection, machine learning, or statistical analyses. All have one in common: Nothing is 100% secure. Reducing the attack surface is essential, it increases the effort for an attacker with regard to time and financial costs. However, threats such as zero-day exploits, vulnerabilities in hardware [80, 72], vulnerabilities in protocols [155], social engineering, or just an upset employee with appropriate privileges are serious issues. There are many current trend reports for cyberattacks available which all draw the conclusion that threat actors have improved their techniques and their attacks have become more sophisticated [45, 156, 86, 62]. Outsiders, like organized criminal groups or state-sponsored actors, perform most of these recorded attacks or cybercrime, but also insiders (malicious or inadvertent) are serious threats in specific industries, for instance, in financial services. In 50% of all breaches, malware was involved and a significant amount (66%) was installed through malicious email attachments. Humans are still one weak point for IT Security and the trend is currently rising. Hence, social engineering and weak passwords are still major threats to system security [156, 62, 37].

Considering these facts, it is perspicuous that threats can be mitigated but not entirely eliminated. Therefore, additional security measures have to be used to monitor and possibly detect attacks or security breaches. When an attacker breaches into an internal network, security measures such as IDS, SIEM tools or honeypots can detect his activities. While IDS and SIEM tools permanently analyze resources, honeypots are passively

waiting for a connection that is by definition unauthorized. Consequently, a honeypot can also capture unknown threats such as zero-day exploits. This is a significant advantage which makes it highly valuable not only for detection but also for research purpose [10, 51]. Using a honeypot enables possibilities beyond detection of an attacker, for example, monitoring his activities in real-time and recording his exploits, tools, and behavior. This leads to a better understanding of adversarial Tactics, Techniques and Procedures (TTP) with the positive addition of advanced information gathering, which could be used for recognition, attribution and in the best case results in an identification.

1.1 Motivation

Targeted, complex and sophisticated attacks, such as Advanced Persistent Threats (APTs), are serious challenges. These threat actors are highly professional criminal or state-sponsored groups. Often they are specialized in lateral movement and covert data exfiltration; they are able to develop own (zero-day) exploits and advanced software/malware. Their TTP includes the compromisation of a system and gain permanent access. Such an attack is adequately planned, well-financed and hard to detect.
During this thesis, an attack on the German government network became publicly known [96]. In contrast to most attacks, the attacker was able to compromise parts of the network. It is known that foreign intelligence services suggested that a network security breach happened. An official statement declared that the attack was isolated, controlled and monitored [22]. Exact details are currently not available to the public but apparently vulnerabilities in the out-dated learning platform “Ilias” were exploited [73]. However, the defenders were able to attribute the attacker and determine the geographical origin, which was Russia. Presumably, the group “Snake”/“Trula” [8, 143] is responsible for the attack, even though APT28 also known as “Sofacy Group” or “Fancy Bear” were mentioned at the beginning [161, 52, 163]. This elucidated how difficult it is to perform an attribution on sophisticated attackers, even for professionals within several months.

Since sophisticated attackers are able to penetrate even highly secured networks, it is necessary to deploy defense in depth. Honeypots are predestined for a layer deep in the network where they can passively wait. They can be used to detect such an intrusion and furthermore, they can collect valuable data. If this data is gathered and processed appropriately it can be possible to recognize the attacker and trace him over several sessions. This can help to identify threat scenarios and threat actors and in the best case could lead to an attribution. At least, the approach will reveal unique repetitive characteristics of a successfully recognized attacker which can serve as Indicators of Compromise (IoCs) for threat intelligence. This could be useful for other potential victims and of course investigating authorities.

1.2 Problem Description

Bruce Schneier describes detection as more critical as prevention since it is impossible to prevent all attacks [120]. According to his description, three categories can be identified in a security process: Prevention, detection, and response. Each of these categories is essential to mitigate and limit attacks and their impacts. The example from section "Motivation" clearly indicates that even accurately devised network perimeter security cannot be used as an exclusive defense measure since it cannot prevent an intrusion [158]. Instead, it adds additional security layers which identify suspicious activities and detect an attacker or malicious insider.
However, by reviewing today's detection methods, it becomes apparent that these are almost exclusively made for detection of known threats. For instance, massive amounts of system logs or traffic data are actively analyzed and validated against a rule set or learned behavior. This is a valid approach to handle known attacks but is insufficient for sophisticated attackers and new attack vectors. Moreover, after a successful compromisation, most of them do not support a reaction or recovery phase. The crucial evidence is hidden in massive amounts of data which leads to slow or incomplete investigations.
Honeypots are a well-known instrument to counter these problems. Due to their passive nature, only high-value data are collected. For this reason, they are predestined for the approach of the thesis. The idea is to detect attackers and furthermore, to subtly influence him so that he can be traced over several sessions. This leads to the following hypothesis:

> *A recurrent attacker cannot only be detected but recognized and tracked through multiple sessions by using appropriate measures based on honeypots.*

1.3 Goals

This thesis is primarily focused on the recognition of attackers. Recognition clarifies the impression of the current attack and attacker and furthermore, allows a better screening of the threat scenario. This can lead to an appropriate countermeasure for a specific threat actor. Since the approach is based on honeypots, which should be as similar as possible to productive systems, the findings are highly sensitive for the corporate network. It is possible to gain IoCs for threat intelligence and possibly features for an attribution. Though, attribution features (more general) are stronger than recognition features (more specific). Therefore, any strong attribution feature is also a strong recognition feature where the inversion does not apply. For example, a unique knowledge in a specific context (e.g., characteristics of a network or passwords of a user account) can result in recognition but not in an attribution.
Also, practical relevance is a necessary condition for this thesis. The chosen approach has to be complex enough to detect and track an attacker, at the same time it has to be simple

enough to integrate into nearly arbitrary infrastructures. The structure is required to be modular in order to be as flexible and extendable as possible. Therefore, an additional security layer based on honeypots will be developed to meet all these requirements.
Finally, it should be possible for nearly any potential attacker to compromise or at least to interact with the honeypots. Several traps will be used to track all activities, collect all data and aggregate them to specific profiles regardless of their skill level. Since a certain skill level and interaction is necessary to allow a valid recognition, targeted attacks by sophisticated attackers will be focused. Passive profiling, as well as active measures, can be used to optimize a recognition. Thereby, the security level for all uninvolved systems should not be reduced or threatened.

1.4 Limitations

Time and finances are two main factors which limit any defender. Of course, this thesis is even more constrained. Therefore, the proof-of-concept is realized with Virtual Machines (VMs) only and without any commercial product. Also, the Internet Protocol (IP) address is in the address space of the TU Dortmund University which is part of the *German Research Network* and might deter (sophisticated) attackers.
Moreover, the evaluation is done by security professionals without a destructive or criminal nature. Their methods and motives can be fundamentally different from real criminals or attackers. For apparent reasons, the approach could not be tested with APT groups but with random real-world attackers. Unfortunately, these attackers did not have the required skill set or willingness to compromise the research setup successfully. Consequently, the significance of the results cannot be generalized.

1.5 Thesis Structure

The remaining chapters of the thesis are structured as follows. In chapter 2 the essential theoretical basics are introduced. Firstly, the concept of honeypots and secondly, the process of attribution is presented. For each section, certain distinct types and characteristics are discussed to show how they are contextualized in the developed approach. Chapter 3 gives an insight into the methodology for a generic approach. At first, the constraints and parameters for the attacker model and the evaluation are set. Then, an entire architecture model including a possible solution for a corporate network and the general concept of the approach are introduced as well as particular systems and measure which are necessary to achieve the recognition of an attacker. Finally, another approach is discussed which was evaluated during the thesis but discarded. Chapter 4 is about the development and implementation of the concepts compiled in the methodology. The research setup and the individual modifications and implementations, which are required for

realizing the approach, are presented. The emphasis is on the topology, the development of a high-interactive Linux honeypot and the transparent proxy. Chapter 5 is dedicated to the evaluation of the chosen approach. Firstly, a group of known attackers and secondly, random real-world attackers are evaluated. Certain general preconditions and parameters apply to both groups which are introduced in each case. Finally, chapter 6 concludes the thesis and discusses the key hypothesis.

Chapter 2

Theory

This chapter gives a brief introduction to the theoretical background for this thesis. Honeypots are presented at first, then the concept of attribution is proposed. Both topics form the basis for the developed approach, and a solid understanding is useful for the later process. Of course, it is not possible to discuss both topics in every aspect and shape, but the foundations concerning the approach will be described sufficiently. Furthermore, an attempt was made to capture the state-of-the-art of both concepts and reference several information resources for further research.

2.1 Honeypots

This section introduces the vast field of honeypots. These explanations are not exhaustive but should build a solid understanding of the basics needed for this thesis. No special cases are considered, such as client-honeypots [53, 112]. Instead, the use case for the presented approach is staked out, and the methodologies and benefits are motivated.

2.1.1 History and Definition

For three decades the methodology of honeypots has been publicly known. The first documents from the early 1990s [131, 15] do not refer to them directly, but the described functionality formed the basis for today's honeypots. Later in 1998, Fred Cohen introduced his "Deception Toolkit" [17] as the first publicly available honeypot solution. His idea was to give the defenders advantages over attackers by using deception for counter attacks. From then on, honeypots have been continuously evolved and improved to counter automated attacks and, at least at the beginning, to trap and analyze worms. A vast amount of research is provided by the Honeynet Project [140], which is a non-profit group of security researchers and is dedicated to honeypots in all their types.
For a long period of time, there was no suitable and straightforward definition of honeypots because of different points of view [109]. Any source of information introduces an

individual interpretation. Today's literature often refers [14, 18, 115] to a definition given by Lance Spitzner, the founder of the Honeynet Project [140]. In his pioneering book "Honeypots: Tracking Hackers" [125] he defined a honeypot as follows:

> "It's a security resource whose value lies in being probed, attacked, or compromised."

This definition will be used in this master thesis. It is a broad and flexible definition which covers a vast field of applications. A honeypot is not necessary a computer system, even if it is often assumed. It could be any resource which has no production value but "in being probed, attacked, or compromised". For example, a router, a mobile phone, a database entry, a file on a computer system, or an entire network. *Honeytokens* [124, 147] are such an interesting form which could be seen as digital or information resources, for instance, files. It is only important that a honeypot simulates the usual behavior and appearance of a real resource which seems to be part of the system/network. Moreover, based on its characteristics, it is essential that it is isolated and entirely monitored. This definition is generic enough to fit several types of honeypots without the need to specify every use case which leads to a confusion of non-technical readers.
A honeypot has no productive use and should not interact with any production system. Therefore, in the context of hardware resources like computer systems, any traffic to or from this system is suspicious. If the honeypot is in the form of data or information like a database entry or a file, any access should trigger an alarm. By offering these features, a honeypot adds an additional layer of security but cannot be used as a defensive measure. Instead, it is part of an IDS where any detected activity involving it is at least a suspicious or even malicious activity. Generally, the false positives or false negatives of an IDS are significantly higher than corresponding metrics of a honeypot, for instance, because of anomaly detection through heuristics. Furthermore, the different modi operandi are decisive: On the one hand, an IDS has to deal with lots of traffic generated by production systems, on the other hand, a honeypot is only penetrated by suspicious or malicious activities. Beside these false alerts of an IDS, misconfiguration and unknown attack vectors affect the detection rate negatively. A honeypot in contrast can be seen as a decoy which lures a potential attacker [138, 21]. It has to be attractive for an attacker to draw his interest. This could be done by responding to port scans or, even better, offering several vulnerabilities like weak passwords or exploitable software. It should be kept in mind that even if a honeypot attracted the attacker's attention, it is not guaranteed, that he is not probing or attacking the production systems. With today's offensive tools it is simple to parallel and automate scans and attacks to pick some low-hanging fruit. This is useful knowledge which helps to find suitable deployment places. The ways where and how a honeypot is deployed is crucial because a honeypot which is not "probed, attacked, or compromised" is worthless.

Another benefit is based on the non-productive characteristic of a honeypot: After a compromisation of a computer system, the affected victim most often prioritizes the availability of a service or a production system over a forensic investigation. Therefore, the compromised systems are prematurely cleaned or formatted and recovered from a data backup. However, if evidence is not preserved, analysis and accounting of the incident are nearly impossible. Offering the attacker the possibilities to exploit the same or maybe further systems in the same manner, using the same unrevealed techniques and vulnerabilities. In contrast, a honeypot can be disconnected and investigated without affecting the production systems and services. Giving defenders, Computer Emergency Response Teams (CERTs) and investigators precious time and insights into the attacker's methodologies and tools. Furthermore, it is possible to correlate the logs from the honeypot and other systems like IDS, firewalls, and proxy servers. With the knowledge gained, existing technical defensive measures can be enhanced to counter this attack and (zero-day) exploits. This improves the security which should be the main objective in accordance with the principle: "Prevention is ideal, but detection is a must." [91]

2.1.2 General Advantages and Disadvantages

The concept of a honeypot has several advantages and disadvantages. These are related to the fundamental methodology which is independent of the degree of the interaction and use case. Lance Spitzner outlined structural characteristics [125], but these can be extended.

Advantages

Data Collection. Honeypots are not part of the production systems, they do not communicate with them, and they do not offer productive services. Therefore, there is no or at least heavily limited legitimate traffic (for example broadcast messages) to or from them. Any connection or traffic is suspicious and potentially malicious. Since they do not offer services, all data on the honeypot can be seen as static. Thus, filtering is either not necessary or quite simple. All collected data are high-value data with negligible noise and barely or no false positives.
Resources. Based on the excluded communication with production systems, traffic overloads which affect (Network) IDS are eliminated. No possibly necessary packet is dropped, and an exhaustive collection is enabled. It is not necessary to inspect and analyze huge amounts of data as the relevant ones are targeted.
Simplicity. There is no need to develop (complex) algorithms or to generate new signatures or rules. A deployed honeypot merely waits passively for a connection which is by nature suspicious.

Quality Assurance. In the event of a compromisation, data and logs of implemented security measures like IDS and firewalls can be correlated. If there is no detection, the systems need to be adjusted to counter or at least mitigate the attack vector. This increases the level of security and enables an immediate reaction to unknown exploits/attacks [122].
Encrypted Environment. Since there is no need for a honeypot to inspect the network traffic, the entire environment can be encrypted. The traffic must not be intercepted, and the encryption must not be broken.

Disadvantages

Singularity. A honeypot which is not "probed, attacked, or compromised" is worthless. If an attacker avoids a honeypot, the compromisation is not detected because no other systems are monitored by the honeypot. Therefore, the amount and placement of honeypots are essential.
Fingerprinting. It is possible for an attacker to identify a honeypot (subsection 2.1.5) for example well-known honeypot-solutions have unique characteristics which can be observed. If an attacker knows that he is dealing with a honeypot he could avoid it, provoke false positives by spoofing production systems or, even more dangerous for Research-Honeypots, feed false information.
Enhanced Risk. Depending on the honeypot, it could be used to start further attacks on other (production) systems or be used as an access point for lateral movement. Isolation and comprehensive monitoring are crucial for these kinds of honeypot. So as not to threaten other vital systems or the entire network.

2.1.3 Types of Honeypots

In 2001, Marty Roesch (creator of Snort [117]) suggested in an article from Lance Spitzner [123], that there are two types of honeypots: Research-Honeypots and Production-Honeypots. Further, according to Iyatiti Mokube and Michele Adams [93], honeypots can be grouped according to their level of interaction. Both models classify the objective of a honeypot. Roesch's model is simple: A *Production-Honeypot* is used to secure an organization's environment [157]. It mitigates risks by detecting and tracking a potential attacker. Therefore, the main objective is to detect an attack. In contrast, *Research-Honeypots* are utilized to study threats and decrease the information gap between the black hats and the security community. Their main objective is to gain new knowledge of the enemies' methodologies, exploits, and tools.
However, the model of dividing the honeypots by their level of interaction is more common. The honeypots are classified as Low-, Medium- and High-Interaction Honeypots. Medium-Interaction Honeypots are ranged in between Low-Interaction Honeypots and High-Interaction Honeypots, their advantages and disadvantages are directly derived. No

detailed declaration on this kind of honeypot will be provided because they only enhance Low-Interaction Honeypots. For example, not just a connection to a service or application is feasible. Moreover, a limited interaction like sending commands or downloading files is possible. Since all these interactions are limited, they do not offer a sufficient enhancement for the presented approach.

Low-Interaction Honeypot

A Low-Interaction Honeypot only offers emulated services or applications. Hence, the level of interaction and the possibilities of an attacker are very limited. These kinds of honeypots can be implemented through simple scripts which offer the ability to establish a connection to a specific service. If desired, the functionality can be extended to allow a file upload or other usual activities provided by the emulated service or application. This could be useful to collect malware such as worms or to detect automated attacks. With these capabilities, it is impossible to exploit the simulated vulnerability, and consequently to infect or compromise the honeypot. The main objective of these honeypots it to detect an attack, not to gather further information about how the attack proceeded. Publicly available solutions often come with monitoring capabilities like logging or malware sample capturing, for instance. Also, honeytokens or other data/information used as a honeypot are considered as Low-Interaction Honeypot.
It is nearly effortless to set up and maintain such a honeypot. Furthermore, the threat level and security issues are not significantly increased. However, the limited interaction may cause an attacker to uncover the honeypot or lose his interest.

High-Interaction Honeypot

A High-Interaction Honeypot provides real (vulnerable) services or applications; nothing is emulated. The attacker can fully interact with the software and Operating System (OS). Therefore, it is a complex solution with immense potential. The gain of an extensive amount of information on one side, on the other side enormous risks. For instance, it is possible to supervise any action performed by the attacker, acquire insights from zero-day exploits (unknown vulnerabilities), new tools or malware, lateral movement and characteristic behavior. This enables a detailed picture of how an attack progresses if there is exhaustive monitoring. On the downside, the attacker deals with real vulnerabilities and, therefore, can indeed exploit them. He can compromise the system and maybe escalate his privileges. If not isolated correctly, the honeypot can be used as a proxy system or access point into the network or it can be used to attack further systems. Hence, it is essential to separate the production network and the honeypot. It is necessary to ensure that the honeypot is wholly monitored and under control, at any moment.

Since it is a complex system, the deployment and maintenance take a lot of time and effort. Also, High-Interaction Honeypot can increase the threat level and may introduce further security issues.

2.1.4 Deployment

As mentioned before the deployment of the honeypot is essential. The way how and where the honeypot can be accessed decisively determines the success. It must be easily reachable to fulfill its task, but at the same time it must be separated or isolated so that it does not provide any threat to other systems. Depending on the level of interaction, certain assumptions can be made. The following description is oriented at Anand Sastry's and Robert McGrew's articles [119, 87].

Low-Interaction Honeypots are running on a host OS and therefore require special software as well as a specific configuration to emulate a vulnerable service or application deceptively authentic. Publicly available Low-Interaction Honeypots often provide fully operational solutions. Monitoring capabilities for the particular service or application are integrated out-of-the-box. Therefore, the integrator only has to configure the honeypot and does not have to deal with logging, sample capturing or real-time notifications functionalities. Some solutions also offer the possibility to extend the out-of-the-box capabilities by installing further add-on modules.
There is no extraordinary effort needed to secure the Low-Interaction Honeypots since all provided services or applications are emulated. The threat level will not be increased significantly. This kind of honeypot is specialized in detecting attackers, automated attacks, and malware such as worms. The very limited possibilities pose no threat to other systems so that these honeypots can also be used in a production network. However, strict monitoring is recommended.

High-Interaction Honeypots have many requirements for secure use. They can have the same level of security as the production systems or provide more vulnerabilities - depending on the use case. One appears to be a low-hanging fruit; the other represents the current security situation. Since nothing is emulated, the attacker directly interacts with the real system. Any vulnerable application or OS version can be exploited. The system will be compromised which is also desired. Two possibilities are distinguished: First, the exposed system runs directly on bare hardware. Second, the vulnerable system runs in a guest OS on the host OS, for instance in a VM. The first approach does not have to deal with Virtual Machine evasion and is quite difficult to detect. But it is costly to implement multiple High-Interaction Honeypots on separate systems/hardware to increase the probability of a compromising. However, virtualization technologies like VMs and

cloud computing are standard in today's operational network [160]. Therefore, a solution based on virtualization is suitable. The following requirements and measures are divided into host and guest OS, but it should be straightforward to relate this to a complete solution without virtualization.
The High-Interaction Honeypot is located in the guest OS. The host OS should be isolated from the honeypot as much as possible. That is crucial for the host system. An attacker who escapes the VM can compromise the host and use it as mentioned before, for instance for lateral movement or to attack further systems. Escaping a VM is not trivial but it is possible [16], and therefore it should be expected or taken into account. To mitigate these risks, the tailing monitoring and protective measures are recommended.
The host OS monitoring should focus on incoming and outgoing traffic concerning the honeypot. Well-known traffic-capturing tools, for instance, Wireshark [146] can be used to achieve this. Also, a firewall or iptables must be used to block or filter outgoing traffic/-connections, and an IDS like Snort [117] should be used to provide alerting mechanisms for known attack vectors.
The guest OS (High-Interaction Honeypot) must be monitored exhaustively. Any keystroke, command, file or behavior has to be tracked and preserved. This enumeration is not sufficient; it is genuinely essential to monitor any activity an attacker can perform. However, this depends on the honeypot and needs to be customized. In general, any read, write and execute operation is in the scope as well as exploit or privilege escalation attempts.
Last but not least, the High-Interaction Honeypot has to be separated and isolated from the production systems. If available, a dedicated or separated network should be used to mitigate the threat level as much as possible.

Both types of honeypots serve different requirements. However, both have in common that secrecy is essential. If an attacker knows that there are honeypots deployed, he will act as silently as possible to avoid them.

2.1.5 Evasion

Honeypots provide many benefits, which are described in subsection 2.1.2. However, honeypots have also become the focus of attackers, for example through their use in companies and government networks as well as media-effective lectures by well-known security professionals such as Haroon Meer at Black Hat USA 2015 [89]. What adversaries know exactly and what countermeasures they have developed is unknown. Though, at least the current state-of-the-art of the research and security community should be assumed.
The scientific research is dedicated to the issue of how a honeypot is identified [66, 58, 19] or even how a honeynet/honeypot can be attacked [27, 136]. Mostly, individual characteristics of the specific honeypot solutions are exploited. The logical consequence was

the development of tools that simplify and automate these processes. Like many other offensive tools, these are intended primarily to serve the security community or, more precisely, the penetration testers. Examples are *HoneypotBuster* [67] and *Kippo detect* [94]. Both tools are simple scripts that take advantage of specific characteristics. This makes it relatively simple for defenders to adapt and deceive the tools. *Kippo detect* is a Python [111] script that sends a sequence of eight "newline" characters ("\n") and looks for the hard-coded value "168430090" in the response. *HoneypotBuster* is a PowerShell script that also checks hard-coded values but being much more complex and "smarter". For example, injected credentials are detected or poorly faked user or admin accounts are uncovered. For instance, the properties "logon", "logoncount" and "lastlogontimestamp" are checked. The tool is based on known honeypots and their specification: "HoneypotBuster is a tool designed to spot Honey Tokens, Honey Bread Crumbs, and Honey Pots used by common Distributed Deception vendors." [67] Also, *Shodan* [85], a search engine for devices which are accessible from the Internet, provides a variant for the detection of honeypots [121]. Like all the tools and papers presented above, they are intended to highlight the vulnerabilities of honeypots and thus awaken an awareness. Last but not least, there are also public discussions about the possibility of identifying honeypots [65].
It must be assumed that all known tools are available to potential attackers, as well as self-developments and advancements. Especially known and widespread honeypots could be identified and avoided. But there are restrictions: For example, *HoneypotBuster* can only be used on the honeypot and thus the attack is at least detected. Also, an identification by *Kippo detect* can be easily detected as an attack. For attackers and their custom tools, however, these observations do not apply. So, there will always be a race of defenders and attackers. As a conclusion, honeypots need to be continually improved/adapted to make identification and fingerprinting as challenging as possible.

2.2 Attribution

Attribution is a well-known concept in psychology and marketing. Today, journalists, investigators, and intelligence agencies are familiar with it as well. In general, it can be defined as:

> "Attribution is the concept of finding the cause for a particular action or object." [75]

This version is an appropriate definition which is suitable for a broad scope. In case of cybercrime or threat intelligence, it can be refined as tracking and identifying a specific threat actor in a cyberattack or cybercrime.
For this section, further terms have to be defined. *Cybercrime* is any illegal activity that involves technological devices and is motivated by personal gain. Formally it is defined as:

> "Offences that are committed against individuals or groups of individuals with a criminal motive to intentionally harm the reputation of the victim or cause physical or mental harm to the victim directly or indirectly, using modern telecommunication networks such as Internet (Chat rooms, emails, notice boards and groups) and mobile phones (SMS/MMS)." [57]

Furthermore, *cyberattacks* (from now on also referred to as *attacks*) and even the attempt of committing a cyberattack are violations of IT security measures or policies:

> "Cyberattack refers to deliberate actions to alter, disrupt, deceive, degrade, or destroy computer systems or networks or the information and/or programs resident in or transiting these systems or networks." [95]

An enhancement of cyberattacks are targeted attacks committed by APT groups. These groups do not only use sabotage, but they are also specialized in espionage. A threat actor is used as a synonym for such a group.
These terms have several intersections but also have some unique characteristics. For instance, identity theft, cyber stalking, cyber harassment, copyright infringement and child pornography are subsets of cybercrime but not included in cyberattacks. Whereas, a Distributed-Denial-of-Service (DDoS) attack is a shared characteristic, even if the motives can be different.

It is evident that attribution is a challenge even for security professionals, investigators, and intelligence. A description as "attribution is an art as well as a science" [114] shows, that it is not just hard facts and science but also experience and soundly based estimates, for instance for the determination of motives.

Cybercrime is highly organized [49], and groups specialized in APTs, as well as intelligence agencies, have extremely skilled security professionals. Attacks performed by such a group are sophisticated and are based on processes. These processes involve an entire infrastructure and are not realized by a single individual [132, 108, 54]. Attribution is an approach to illuminate these activities by answering various questions regarding their threat processes: Who designed/planned, organized and performed the attack? Who wrote the malware/software, built and provided the infrastructure, for instance, Command-and-Control-Server (C&C-Server)?

2.2.1 Attribution Process

Any cybercrime and any attack leaves marks such as a (spear-) phishing email, a malware sample, a compromised website, or various log files. Even if an attacker tries to leave as few indications as possible or tries to erase them. However, the linkage of a single attack to a specific threat actor is quite difficult. Therefore, preparations have to be made to allow a successful attribution. This process can be described in five different steps [129].

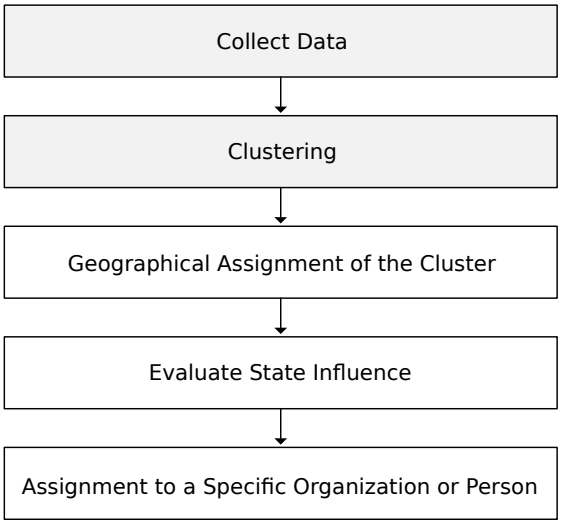

Figure 2.1: Schematic representation of the attribution process

Figure 2.1 describes these steps schematically. The first two steps (highlighted in gray) are independent of the attribution. Instead, they indicate data collection, preparation, and data mining. The last three steps represent the actual attribution. In the following, these processes are described in more detail.

First Step: Collect Data. It is essential to collect as much data as possible from different and well-selected sources. Good sources are, for example, security/antivirus companies, internet providers, large technology companies or authorities with corresponding departments. Of particular interest are sources such as malware sample, malicious websites and emails, C&C-Server, log files of compromised systems, and custom-made tools.

Second Step: Clustering. After gathering these data, they have to be aggregated. Depending on the data this could be done in different ways. However, they all have in common that they have to be similar, corresponding to comparable characteristics/features. Such a cluster is an abstract representation of retaining data. For instance, malware samples can be aggregated into families or attacks, while spam may be aggregated into campaigns. For this purpose, characteristics are examined for similarity and inconsistencies are resolved. In addition, the characteristics are weighted differently. The severity of a feature is determined by its uniqueness and possibility to forge it. Authentication methods (complex password, private keys) and custom-made malware, for example, are strong features. Weaker characteristics are, for example, same kinds of victims, same IP address range, reuse of publicly known source code.

Third Step: Geographical Assignment of the Cluster. At first the potential beneficiary of the attack is under suspicion. In addition, victimology is a useful but not sufficient condition. An internationally active company, and especially governments, can have various adversaries. Therefore, digital forensic can help to restrict the group of perpetrators. For example, malware/software, protocols, or patterns of life can leak valuable information like language settings or geographical particularities, for instance, special holidays or time zones [128]. As mentioned in the second step, data has to be collated and inconsistencies have to be resolved.

Fourth Step: Evaluate State Influence. After assigning the cluster to a geographic area, the origin is delimited. First, it is decided whether there has been any state influence or organized cybercrime. For instance, the number and distribution of victims, the functionality of software (espionage or theft of credit card information), or the complexity of attacks (watering hole attack, spear phishing or mass spam) are evaluated. In addition, motives and interests are considered.

Fifth Step: Assignment to a Specific Organization or Person. The last step succeeds very seldom: An attack is linked to a specific threat actor such as an intelligence agency, an APT group or a group of organized cybercrime. Ideally but very rarely a person is identified. This occurs only when gross mistakes are made [88, 20].

2.2.2 Types of Attribution

According to this process, it is possible to subclassify attribution into relative and absolute attribution [75]. Regarding this sequence, relative attribution can be seen as an almost complete run including the assignment to a defined but abstract and not identified group, whereas absolute attribution identifies the exact person behind an attack or cybercrime. For this thesis the following definitions are used:

Absolute Attribution

The objective of absolute attribution is to determine, who exactly performed an action or, in this setting, who performed a cybercrime or attack.
Apparently, this level of attribution is hard to achieve. Due to highly skilled and organized adversaries, practically only an acknowledgment or gross mistakes can give assurance. A good example is Stuxnet where many assumptions were made, evidence for a specific attribution and evidence to the contrary were presented, but finally, only U.S. President Obama's acknowledgment brought clarification [38, 29, 106, 118].

Relative Attribution

Relative Attribution has weaker conditions: It will not link to an identified perpetrator but a defined group or individual. The main objective is to link two or more cybercrimes or attacks to the same threat actor. The identification is not included in the relative attribution, it is not the purpose to determine who the attacker is. Good examples are most APT groups; except for suspicions, nearly nothing is known for certain or with the requisite level of confidence. Particular pattern or even characteristic signatures indicate their unique methodology and strategy. Information gathering from multiple incidents and/or authorities can create a profile, which represents such a group or individual.
In case of attribution, authorities usually refrain from giving definitive statements, only in sporadic cases names are mentioned [152, 151]. Most times they only refer to a country, while the governments concerned deny anything. "NotPetya" represents a current reference case, while investigations are still ongoing [46, 145, 137].

2.2.3 Reasons for Attribution

Cybercrime or complex attacks are performed by organized groups as mentioned before. Classification of such an attack can help to mitigate the threat for other individuals, organizations or even governments. Furthermore, it can create a better picture of the attack, the motives, and the threat actors. Linkage of evidence and other gained information, for instance, IoCs, can reveal an entire campaign of attacks which is not only targeting one organization but a whole industry or even critical infrastructures like water supply. This

knowledge can be used to prevent, detect or at least mitigate attacks of the same threat actor. If a campaign of attacks is identified, the awareness increases and more resources can be used to repel the attacks. Of course, the evidence and findings can also be used for further investigations.
Also, it helps to prioritize countermeasures by threat level. Since time and finance are limited, it could be necessary to order threats by the likelihood of becoming a victim. Threat actors are specialized in individual domains, for example, specific industries or governments, and certain methods, like espionage. Therefore, it is advisable to prioritize the respective domain and their threats/threat actors.
From the viewpoint of an attacker, attribution has to be countered. This forces him to continuously change his pattern, techniques, and methodology or to mimic another threat actor. However, this leads to a more significant effort for time and costs. Alternatively, a generic attack could be performed by using publicly known exploits and tools, which could be simpler to detect and counter.
Furthermore, if he has to expect to be exposed and maybe to bear the (legal) consequences, then his willingness to participate in such an activity can be reduced.
At the diplomatic level, it can also be used by governments to sanction influences by other states. Or rather, lead to an agreement of coexistence and collaboration [144].

2.2.4 Challenges and Limitations of Attribution

As mentioned before, attribution can affect how threat actors behave if they are conscious of this process. Organized cyber criminals and skilled attackers, like APT groups, know what they are facing. Therefore, significant effort is put into avoiding attribution or even to mislead investigations. If the attribution is incorrect, due to inadvertent analysis or deception of perpetrators, the linkage of information and evidence is false and could lead to a false positive. It is a basic technique of APT groups and intelligence services to set wrong tracks. Some attempt not only to distract from themselves, but also to blame others by performing *framing attacks* [3, 7, 55]. For example, usage of tools or code fragments of other groups, or false flags can be placed. Moreover, it can be pretended to be in a different location by using certain timezones, patterns of life and corresponding language settings as well as spoofed IPs.
Due to anonymization techniques, it could be difficult or impossible to assign an attack to a state correctly. Virtual Private Network (VPN), proxy server, compromised systems or botnets can be used as a stepping stone and also lead to a false attribution. For instance, such a compromised system can be used for *multi-stage attacks*, where the system of the first victim is used to attack the target. The “MuddyWater” campaign, from 2017, and a possible connected new campaign, from 2018, are good examples for such an attack and

attribution of similar campaigns: “Hundreds of hacked websites are used as proxies” and false flags are used to hinder the attribution [59].

Figure 2.2: Simplified representation of a multi-stage attack

Figure 2.2 outlines a multi-stage attack. The form shown is the simplest variant with only one system/victim as a stepping stone. However, the victim “Stepping Stone” can be as complex as desired, both in the vertical and in the horizontal extension.
These techniques, mentioned above, are known as *Operational Security* (*OpSec*). For technically advanced adversaries it is a standard operating procedure to hide their activities. Since there are publicly accessible documents due to research [142] or data leaks [162], it is straightforward to implement these as a process stage before launching the attack.
Finally, the cooperation of blamed states is often refused. Even if the participation is strictly denied, the accused threat actors are treated as citizens and therefore are rarely extradited or persecuted [139].

2.2.5 Attribution Techniques

In the concept of attribution, an attack is broken down into separate threat actors, which are individually targeted. For instance, a malware/software sample is linked to its writer by attributing characteristic information. These could be the programming language, the coding style, the keyboard layout, the used compiler with parameters, the compiling time, and further more. This is one kind of authorship attribution which has been researched in various forms [4, 47, 164, 99]. Other stages of the process have different, individual links. The most challenging part is the client/initiator as any involved party adds its features and impacts. Therefore, the motive or the expected beneficiary must be determined and in some cases assumed as an educated guess.
The foundation for attribution is always a large amount of data. These have to be analyzed to discover connections and correlations. For this purpose, common techniques from statistical analysis and data mining are used. The outcome should be a model which could be tested and verified. On this basis, the data can be examined to find further similarity and inconsistencies, which have to be resolved first, by using inductive and deductive methods.
But separately analyzed attacks have unique characteristics which are suitable for attribution as well. Any attacker or any group has a specific behavior. This is reflected, for example, in the usage of favored tools and tactics. Also, it could be a preference for

watering hole attacks, social engineering techniques, or specialized software. Sometimes even the order of activities (besides the classic cyber kill chain) or used tools can represent a unique fingerprint. Especially custom-made software and malware are appropriate characteristics to recognize an attacker.
Furthermore, the forensic evidence such as Meta-information, language settings, protocol configurations, source IP ranges, C&C-Server, and domains can be analyzed to extract additional information. Unlikely, but valuable are usernames, acronyms of developers and email addresses. All this can be used for further investigation and attribution, while the last features can lead to an absolute attribution.

It is crucial for attribution to consider some principles to avoid inaccuracy: The validity of the evidence should be evaluated. Consistency must be guaranteed over several sources. Any inconsistencies may indicate manipulated evidence, false tracks or corrupt data and must be resolved. The identified characteristics/features must be weighted according to their significance and reliability. And finally, alternative hypotheses have to be considered, analyzed and evaluated.

Chapter 3

Methodology

3.1 Attacker Model and Evaluation Period

Three adversary models are considered, which are differentiable in characteristic access options: physical and network access besides a web attacker. Based on the use case of honeypots in an industrial or a critical network, web attackers are not separately treated. For simplicity, this model is focused on Industrial Control System (ICS) or networks, which are used in an isolated (industrial) environment. This assumes that all deployed honeypots cannot be reached directly over the Internet. A web attacker will be included if he can compromise a system, which promotes him to an attacker with network access (staged attacks). As an active network participant, he can use all network services and interact with all reachable clients and servers. This role will be the default adversary regarding this thesis and includes all employees with similar privileges and access to the network. The most powerful attacker has physical access for at least a short period of time to one or more systems in the network. He has insider knowledge and can easily identify and interact with real systems. There is no possibility to track his actions such as copy data to a removable data storage device or disconnect the client with (virtual) honeypots. It could be possible to detect such an attack by using, for instance, honeytokens. However, it is not feasible to recognize him with this approach. Hence, he will be neglected if his activities could not be tracked in the network itself.
Furthermore, the capabilities of an attacker should exceed a simple usage of publicly available tools. The knowledge an attacker gains is compromised with injected information. A certain effort has to be made to get this information. Automated bots or attackers with tool-limited or no skillset will not be able to extract this information. Even if these perpetrators are not the center of the research, the approach must also be suitable for them. Therefore, all their activities must be detected, but they should not be able to pass the first system. This guarantees the security of the network and reduces the number of attacks which are relevant to this research.

Besides the attacker model, there must be a definition and constraint for the period of time observed with this approach. According to the latest report by FireEye, Inc. [45], the average dwell time of an attacker in a network until he is finally detected is up to 99 days. Contemporaneously, a Red Team can obtain access to critical privileges (e.g., domain administrator) or sensitive information in roughly three days. Resulting therefrom, the sustained period of time should be from a few seconds up to at least 100 days. In order to be able to practically evaluate random attackers within the thesis, the evaluation period is limited to one week. This is more than twice the time a Red Team or a sophisticated attacker will need to gain access to sensible data – regarding the report mentioned above. Of course, a Red Team cannot adequately be compared to a malicious hacker: On the one hand, there is a limited scope (fewer options), on the other hand, a hacker has to make as little noise as possible and has to be careful in his lateral movement. Nevertheless, it should be possible to extend this period without significant restrictions.

3.2 Architecture Model

The architecture is designed to handle attackers defined in section 3.1 “Attacker Model and Evaluation Period”. Since the objective is the recognition of an attacker, it is especially construed for human interaction and less for fully automated attacks such as worms or bots.

As mentioned in section 1.3 “Goals”, the architecture has to be complex enough to detect and track an attacker. In addition, it should be simple to integrate into arbitrary networks - with a focus on isolated (industrial) networks. Therefore, the architecture for this approach is modularly organized. This allows a flexible structure and reduces the dependencies for a module to a minimum. As a result, no generalized assumptions and limitations must be taken into account for the entire architecture, instead only those for the individual modules are necessary. These assumptions and limitations can be extended according to the respective use case in which the modules are used.

Figure 3.1 illustrated the general structure of the architecture modeled on a potential deployment in a corporate network. This structure is not optimized for a specific use case, which allows introducing general assumptions and methodologies.

The network is partitioned into four separated subnets. Network access across the subnets is restricted by firewalls. This is a protection against unauthorized network access and can be used to monitor/filter data traffic at the network boundaries at the same time. For example, this prevents direct access to the Internet from the Productive Network and reduces the attack surface, because only the necessary ports and services are accessible. The subnets Internet and Productive Network are not presented in detail, as they are not directly relevant to the chosen approach. The Internet, or in general any public network, is the least trusted subnet. It is expected that most attacks are carried from this subnet.

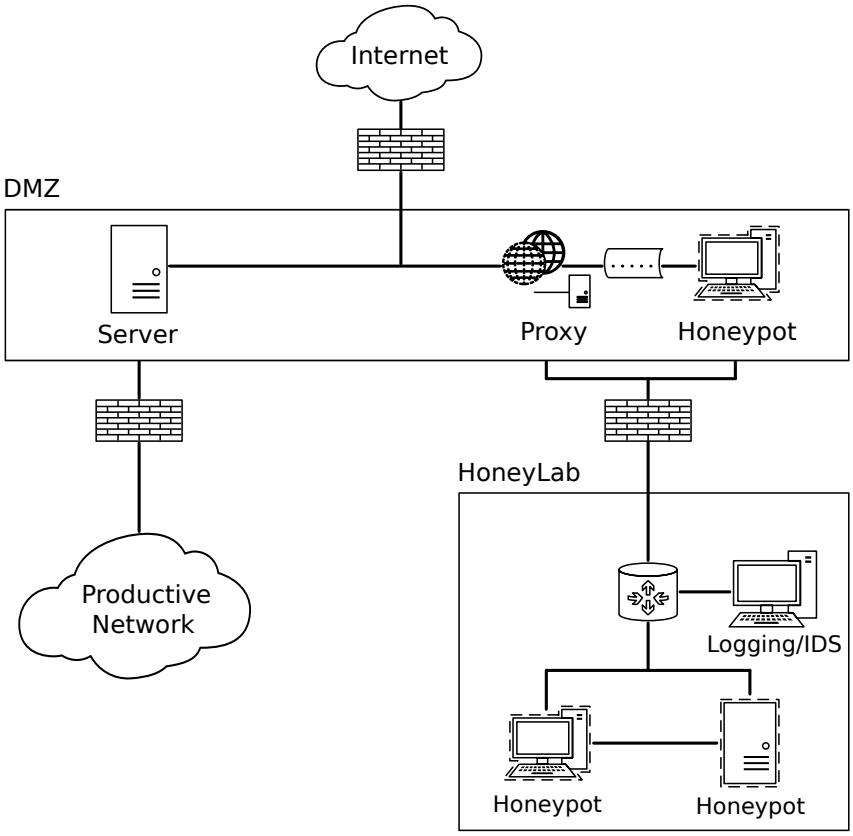

Figure 3.1: General structure of the architecture for a corporate network

The Productive Network, on the other hand, has the highest level of trust. Therefore, it is vital to preserve the connections and internal structure. In this subnet the assets, in form of IT and maybe Operational Technology (OT), are located. For some use cases, it could still be worthwhile to deploy several honeypots in the Productive Network which mimic these assets and have another connection to the *HoneyLab (HL)*. This could be an enhancement of the approach and could be done analogously to the deployment in the *Demilitarized Zone (DMZ)*.

The DMZ is a critical subnet: This subnet separates the untrusted external networks (e.g., Internet) and the trusted internal networks (e.g., Productive Network). Systems located in the DMZ can be accessed from internal and external networks such as the Internet, the Productive Network, and the HL. However, it should be restricted to access internal systems from the DMZ. For instance, for any internal network (e.g., HL and Productive Network) only ingoing traffic on limited ports should be allowed if the connection is initiated from the internal network. Inside the DMZ are common systems deployed such as web and mail servers. Furthermore, there is a transparent proxy as well as a High-Interaction Honeypot (from now on a honeypot is generally referred as a High-Interaction Honeypot, unless explicitly stated as a Low-Interaction Honeypot). The proxy acts as an intermediary to the honeypot, and since it is a transparent proxy, it is hidden from any client.

This allows to eavesdrop the communication between clients and the honeypot. Due to the fact that there should be no legitimate connections to the honeypot, any connecting client could be seen as an attacker.
Since the HL is built to monitor and analyze attackers, specific security aspects of the DMZ have to be violated. It is not allowed to access the DMZ or even the Internet from any system/honeypot located in the HL, but it is allowed to access any honeypot inside the HL form the DMZ. This "misconfiguration" inverted the rules for ingoing and outgoing traffic for the HL. Ingoing traffic has to be allowed, and outgoing traffic has to be restricted to trap the attacker in a secured environment. It should not be possible to use any honeypot inside the HL as a *stepping stone* to enable a multi-staged attack targeting internal or external networks, besides the HL.
The HL can exclusively accessed from honeypots and has no productive use but should give the opportunity to monitor and analyze the attacker. Various challenges and structures are conceivable to induce an attacker to interact with the environment. Basically, it is recommended to use a simple client-server architecture and a monitoring/logging system such as an IDS. It should be a plausible scenario (same level of security and the same software/operating systems) for the attacker, which is similar to the Productive Network. In this case, high-value data for the specific use case and the network can be gathered.

3.2.1 Transparent Proxy and High-Interaction Honeypot

Since the DMZ is located between the trusted internal networks and the untrusted external networks it should contain all systems which need direct access to the Internet. Therefore, it is possible for any client (internal and especially external) to interact with any system inside the DMZ – to the extent permitted by firewall rules. Besides standard servers like mail or web servers, it contains the honeypot in conjunction with the transparent proxy.

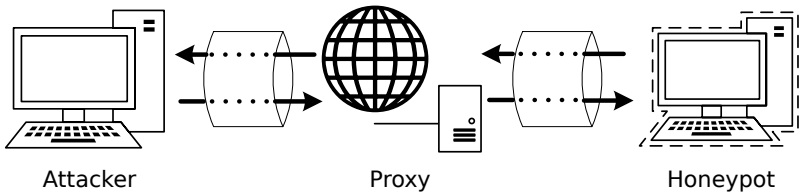

Figure 3.2: Functioning of the transparent proxy

Figure 3.2 shows the attacker's access to the honeypot. While the attacker assumes that he is directly connected to a system, he is in fact connected to the transparent proxy, and the transparent proxy is connected to the honeypot. In consequence, the proxy has to spoof the connections or at least forward the relevant connection parameters. This structure allows offering the attacker an encrypted connection such as Transport Layer Security (TLS) or Secure Shell (SSH). The transparent proxy works as a man-in-the-middle and allows for

eavesdropping, intercepting and even to manipulate the communication. Any command can be logged in plain text, and any file/sample can be copied. This allows complete monitoring and control of the data flow.

The primary task of the transparent proxy is to monitor and collect any activity of the attacker. Moreover, it has to transmit the received data to a centralized logging system, where the data will be stored, aggregated and analyzed. To deceive the attacker, the proxy can spoof the connection from the honeypot and has to support all (simulated) vulnerabilities. Since the influenced knowledge of an attacker is used to recognize him, it is necessary, for example, to offer and change unique characteristics on the honeypot. Accordingly, the proxy or the honeypot itself has to be able to modify specific data on the honeypot without an attacker noticing. The most appropriate timing for this is after establishing the second connection from the transparent proxy to the honeypot, which is actually the logon of the attacker.

It is necessary to arouse the attacker's interest to be able to recognize him. Hence, an environment is needed which allows him to interact and probe several aspects of the honeypot such as services. The honeypot should offer multiple options to interact with in order to have several opportunities to recognize him. Since the capabilities of an attacker are not assessable in advanced, the challenges should differ in the level of difficulty. However, the focus should be on sophisticated attackers.

The recognition capabilities on the honeypot are essential. This system is the entry point into the HL and should lay the foundation for recognition and a reliable attribution. A focus on different strong features, such as exchangeable authentication methods for the honeypot itself or the HL, is advisable. In addition, passive profiling with weaker features such as browser and client fingerprints, time zones, and IP addresses should be used. Over several sessions, this could identify false flags or provide a better picture of the attack and the attacker.

Under controlled conditions, an attacker should be able to change characteristics of the honeypot, like passwords, or even to compromise it. It is unlikely that more than one attacker (or a specific group) is active at the same time, but it should be considered. Therefore, the specific characteristics should be changed every session without the attacker noticing. This requires measures to be taken to keep the attacker unaware. For instance, a private key could be extracted from a configuration file or script. The attacker should not be able to use this key directly by referring to the key file; he should have some effort to extract and implement the key in his tools. By injecting new keys in every session, it is possible for other attackers to get unique keys. At the same time, the "known" attacker should be able to use the old keys and should not be lead to verify his extracted credentials. Consequently, the honeypot has to support various credentials for each user and has to share these with the transparent proxy.

Furthermore, the honeypot has to lead into the HL without being too obvious. The possibility to reach an internal system from the DMZ is a misconfiguration which could raise suspicion by a skilled attacker. It is important to make it seem like a careless mistake. However, the honeypot itself should be enough to get a reliable attribution. Due to suspicion or incapacity of the attacker, the HL may never be touched.

3.2.2 HoneyLab

The HL is an environment which should be as similar as possible to the Productive Network. The infrastructure differs, depending on the corresponding use case. In general, it is possible to deploy any system, but offering access to a server could be tempting for an attacker. This could be a web server as well as a Supervisory Control And Data Acquisition (SCADA) server. The attacker can interact in a certain context because an entire infrastructure is given. It is possible for him to monitor and probe anything in the HL and therefore, nearly any kind of protocol or system can be used. This provides the possibility to learn new adversarial techniques or discover zero-day vulnerabilities.
Besides the client-server architecture of honeypots, there should be a monitoring system which logs and monitors any activity of the attacker. This monitoring system should be state-of-the-art and is used to complement the specific attribution techniques with common monitoring intelligence.
In general, the HL offers many possibilities, such as different operating systems, services and files, to challenge and recognize the attacker. This could reveal his skillset and maybe his motives. For instance, a client can contain files, emails or shared folders from different departments such as financial, research, and management. This could expose his objectives. Depending on time and intention, this can be done as fine-grained as desired.

3.3 Recognition Measures

Once the architecture has been introduced in section 3.2, some approaches are now considered that allow recognizing an attacker. The main concept is to let him build up compromised knowledge. This knowledge should be enriched with strong identification features regarding recognition and attribution, so that it can be used against the attacker. For this purpose, properties of the architecture, as well as further passive and active elements, are used. An active measure is defined as an interaction with the attacker initiated by the architecture. Measures that change the architecture (e.g., the honeypots), for instance, the injection of unique information, are considered as passive methods because the attacker has to acquire the knowledge on his initiative. The active measures can provide further valuable information but must be balanced with the respective law. This includes, for example, the setting of *Evercookies* [69]. This zombie cookie (will be recreated after

deletion) is written in JavaScript [97] and uses multiple techniques to achieve persistence in the browser. Depending on the case-law of the respective country, active measures could be considered a *hack-back* which has legal consequences, for instance in Germany. In addition, experienced attackers are able to set false flags and/or use proxy systems or staged attacks. Thus, uninvolved persons or prior victims could become the target of active (counter-) measures.

Figure 3.3 presents a scenario of an attacker interacting with the architecture. The attacker's interactions are highlighted in red, while the active and passive measures of the architecture are outlined in green. Measures such as logging or securing the copied/downloaded files are not explicitly marked.

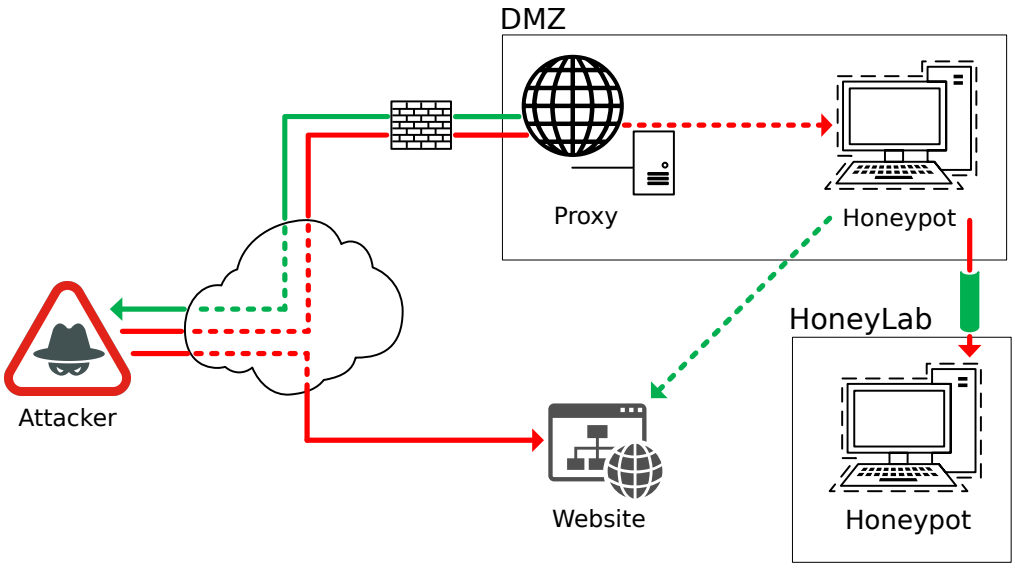

Figure 3.3: Interaction model of attackers with architecture-driven recognition measures

The attacker can access the DMZ over the Internet, although the connection is restricted by a firewall. This affects the attack surface in such a way that only specific ports are available. As a result, the possible attack vectors are limited, and the attacker can be indirectly guided to particular services. Thus, the attacker can identify an intended vulnerability and attempt to exploit it. If the attacker is able to establish a connection to the system in the DMZ, he will transparently be forwarded to the honeypot. The dashed red line illustrates this process. The attacker's assumption should be that he found a vulnerable system and established a direct connection to it.
In fact, the attacker has connected to the proxy, which forwards to the honeypot. Both, the proxy and the honeypot are logging the activities of the attacker. In addition, the proxy performs an active measure after the attacker has successfully logged in: The attacker is

automatically scanned by the system. This scan can include port scans and/or operating system and service detection. The findings are unlikely to result in strong attribution features, but they do reveal parts of the attacker's infrastructure and can be seen as a kind of client fingerprinting. Especially striking/rare combinations of accessible ports can be an additional recognition feature. Also, a port scan is not very harmful to an ordinary system of a potential victim (multi-staged attack).

Two passive measures are carried out in parallel on the honeypot itself: On the one hand, a unique Uniform Resource Locator (URL) is prepared, on the other hand, unique credentials are placed. These characteristics are designed to be strong attribution features, so it is crucial to ensure that this information is unique and not reused. The attacker can obtain and use this information during the session. If these are used across several sessions, there is a significant relationship between the session where the information is placed and the session where the information is used. The attacker can thus be successfully recognized. For the URL, for instance, a Domain Generation Algorithms (DGA) can be used that randomizes the path "reasonably". Common paths or respective word lists can be described as useful. Less suitable are explicit parameters or alpha-numeric sequences, since these are very unusual and can be suspicious. As credentials, dummy accounts conforming to the naming policy schema are beneficial.

Furthermore, a prepared website provides browser fingerprinting [28] as another source of information about the attacker. Various capabilities are available that can be implemented: The request fields of the Hypertext Transfer Protocol (HTTP) header fields [40, 41, 42, 74, 39, 43] can be evaluated, *Evercookies* [69] can be set, and JavaScript and/or further web technologies can be used to detect characteristics such as installed fonts, plugin and cached images. There are several projects [28, 13, 11] in this research field, which are also publicly available. In most cases, this information is not a strong identification feature, but it expands and stabilizes the overall picture. With an unsuspicious website, there could also be the possibility that the attacker visits the bait from various systems. This can lead to mistakes (visit from own system) or reveal further parts of his architecture.

The placed credentials allow controlled access (green tunnel) into the HL. The data must be placed in such a way that the attacker has to make some effort to extract it. This ensures that he builds up knowledge and cannot merely reference the data. For instance, a script that automatically establishes a remote connection is unsuitable because it can just be called by the attacker and, therefore, he does not get any unique knowledge injected. However, if he is forced to invest time in acquiring the knowledge, he will conserve this knowledge so that he does not have to make this effort again for each session. The challenge is that he has to find the credentials, but they do not look like bait.

This kind of access to the HL also has a certain filtering function: The most significant threat, but also the greatest research benefit, comes from skilled attackers. Attackers who do not manage to obtain the credentials or automated attacks such as bots cannot access

the HL. This class of attackers must also be reliably detected, but their analysis does not bring significant additional value to this research. The attackers in the HL, on the other hand, demonstrate their behavior and TTP in an environment that is at best modeled on a productive environment. They are the focus of this research, as they are expected to invest time and extend the attack over several sessions. Their TTP are very valuable for research as well as for defenders and offer the possibility for a reliable attribution.

The approach can be extended by several other measures which are not shown in the picture. For example, honeytoken (various documents), mailboxes that can remotely accessed or an entire vulnerable Content Management System (CMS) that enhance the HL in the form of a web component.

3.4 Discarded Approach: A Specific Protocol Solution

Another approach was considered in this thesis: Modifying specific protocols to inject unique characteristics or even a *honey cookie*.
The *honey cookie* idea is inspired by traditional web cookies [6] which are used to save stateful information. In the context of an attack, the attacker represents the client, and the honeypot is like a web server. The attacker sends a request and initiates the connection. The response of the honeypot contains the *honey cookie*, which is transmitted in any request of the attacker. For example, a unique list of available services/systems or specific session parameters can serve as a *honey cookie*. Unfortunately, this approach has various assumptions and drawbacks. First, not all protocols can be used since it is necessary to have a kind of a state which is stored on the client. This limits the use cases and excludes many Internet of Things (IoT) and industrial protocols. For instance, modern IoT and industrial protocols like Message Queuing Telemetry Transport (MQTT) [5] and OPC Unified Architecture (OPC UA) [100] use a dedicated server (*broker*) to store the states in a dynamic network. A new client and a potential attacker is forced to discover the network and provided services on each session. Furthermore, the attacker has to accept and use the cookie but unlike in the case of web cookies, the attacker has no benefit by sending a *honey cookie*. Without a necessity, it cannot be assumed that an attacker will incur extra effort by accepting and sending a non-standard cookie.
The problem of injecting unique characteristics, for example, in unused header fields which will recognize an attacker can be reduced to similar issues. Unique information such as IP addresses and services will be countered by the discovery stage. On each visit, the attacker will contact the broker to request new information (e.g., available clients and services). The broker cannot recognize the attacker or distinguish between several clients unless the unique characteristics related to the broker himself. Because of the nature of a

broker, it has to be publicly known to enable discovery. Therefore, characteristics such as an IP address or various services have to be unique or for any client and potential attacker. However, this makes recognition impossible since these characteristics are unique for any client and attacker on each visit.
Moreover, if a server is publicly accessible but the supported protocol is such a specific protocol, it will be difficult for an arbitrary attacker to identify and understand the communication and provided services. From an external view, there is no context and no internal communication to analyze. Even if he interacts with the honeypot, there must be an attractive objective such as the possibility to compromise the system or acquire valuable data. Otherwise, he will lose his interest, leave the honeypot and makes recognition unfeasible.
In general, this approach can be valuable because it is possible to analyze an IoT or industrial protocol. Flaws or new vulnerabilities can be discovered [159]. However, it is crucial to give an attacker a context, and it should not be the only option to recognize him. It could be a valuable enhancement to the approach presented above. The HL could be the right place to implement such a protocol in a specific context and various recognition methods.

Chapter 4

Implementation

In this chapter a Proof of Concept (PoC) for the architecture model introduced in section 3.2 is presented. Due to the limited resources for this thesis in regards to time, finances and IT systems/infrastructure, some restrictions had to be made. The actual research setup includes a physical workstation (Processor: Intel Core i7-7700 CPU @ 3.60GHz x 8, Memory: 15,5 GiB), a separate network and a publicly accessible IP address. This IP address is in the address space of the TU Dortmund University which is part of the *German Research Network* and might deter (sophisticated) attackers. However, these constraints do not change the validity of the results, since the modular structure of the architecture allows horizontal and vertical scaling. For instance, the process is not totally automated. Data transmission and analysis are primarily performed manually. Yet, manual or automatic, this does not change the elementary process. Also, it is often sufficient to verify that a specific measure influenced an attacker, for example, by successfully injecting compromised knowledge. The focus is on recognizing an attacker, even if the full potential of individual measures has not been exhausted. Therefore, the PoC includes several approaches of recognition measures which can be enhanced.
Additionally, the HL was abstracted and minimalized. For a demonstration of the capabilities, one further system running a Microsoft Windows OS was deployed. This may, of course, be extended by an entire network similar to the production network. Nevertheless, this should be sufficient to arouse interest and allows casual attackers to gain further knowledge of what can be used against them in the recognition and attribution process.
Figure 4.1 shows the implemented PoC. The key components of the architecture are the systems in the DMZ. These are supplemented by the highly flexible and individual HL, which can be extended and adjusted as required. Systems located there should simulate the production environment and offer additional valuable data about the attacker. However, the foundation is formed by the Transparent Proxy and High-Interaction Honeypot.

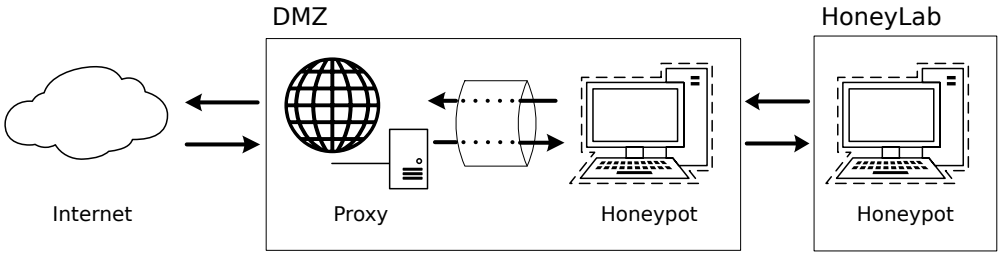

Figure 4.1: Schematic representation of the PoC

In this PoC the Transparent Proxy is an SSH proxy. This meets the requirements placed on the proxy (e.g., authentication and encrypted communication). Furthermore, the SSH service is often scanned, probed and attacked [56, 2], due to the numerous publicly available attack tools and common system misconfigurations. The accessible honeypot was developed from an ordinary Linux system. Since the development of a High-Interaction Honeypot is not the focus of this thesis, this has been simplified. While the general functionality has been implemented, the techniques for obfuscation and data provision have been reduced to a minimum for the limitations mentioned before. For the PoC it should be sufficient to withstand an initial inspection of the attacker. Also, the data provision was severely limited. Captured log files are gathered on the honeypot and subsequently analyzed separately. This reflects the limitations of IT resources and time but is useful for evaluation. As a positive side effect, the different levels of obfuscation and implementation reveal the capabilities of the attackers. For instance, some measures are comparatively simple to identify, although most of them are already covered by the proxy.
The analysis is not performed in real-time or automatically, but manually after the sessions. For a real-world deployment, it can be assumed that more resources are available to monitor the network, even in real-time. In addition, this enables captured data to be provided in real-time using methods or tools already integrated in the PoC, for instance the “Elastic Stack”[36]. Alternatively, the current migration can also be automated by automatically transferring the data after the session or after certain time intervals. Nevertheless, this does not change the validity of the results, since a real-time analysis of the data or an improved obfuscation of the activities do not change any factors of recognition or attribution.

4.1 Topology

The topology of the PoC is designed for a virtual and simplified architecture. However, the concept can be enhanced to accommodate physical and highly complex structures without the need for significant adjustments. The purpose was to develop a holistic approach suitable for any network, which meets the definition from 3.2 “Architecture Model”.

The technical configuration for the research setup is a workstation, a separate network and a publicly accessible IP address. Exclusively IP version 4 is used, and all Network Interface Cards (NICs) [102] have been configured accordingly.

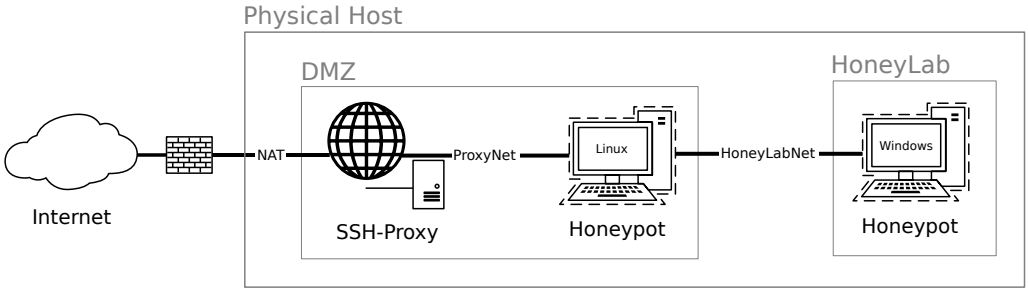

Figure 4.2: Simplified representation of the topology

Figure 4.2 outlines the applied topology. The workstation ("Physical Host") is protected by the firewall of the TU Dortmund University and has direct access to the Internet. In addition, port 22 has been opened to offer SSH on its standard port. The attacker is restricted to such an extent that he can only communicate with the research setup via SSH. Furthermore, he cannot directly access the Internet from any honeypot. The workstation contains the entire virtual research setup. The current version (5.1.38) of *Oracle VM VirtualBox* [101] was used for virtualization.

The workstation was accessible via an IP address of the address space of the TU Dortmund University. To forward the attacker to the SSH proxy, every Transmission Control Protocol (TCP) connection the host is receiving at port 22 is forwarded to port 2222 of the guest system (SSH proxy). Since VirtualBox, in the default configuration, rewrites the source IP address within a Network Address Translation (NAT) mode to the IP address of the NAT gateway, the following command had to be executed on the workstation to ensure that the source IP remains unaltered:

VBoxManage modifyvm <Name of the proxy VM> –nataliasmode1 proxyonly

This is essential because otherwise, the IP address of the attacker is not available in the research setup, e.g., for the active scan and the journals.

The "Physical Host" provides the virtual infrastructure, which is divided into a DMZ and the HL according to the architecture. Within the DMZ an SSH proxy and a honeypot with a Linux OS are located. The HL can be reached via the DMZ honeypot, which refers directly to a honeypot with a Microsoft Windows OS.
In the DMZ, the SSH proxy is accessible via port forwarding. The SSH proxy has two virtual NICs: A NAT connection to the host system and an internal network ("ProxyNet") to the honeypot. An internal network in VirtualBox only allows communication

from connected VMs. A connection between host and VMs as well as between VMs and the Internet should therefore not be possible - at least at the time of the thesis no vulnerabilities or exploits are publicly known. The two virtual NICs are configured as follows:

IP Address:	10.0.2.15
Broadcast Address:	10.0.2.255
Subnet Mask:	255.255.255.0
Default Route:	10.0.2.2
Primary DNS:	129.217.xxx.xxx

Table 4.1: NAT configuration of the HonSSH proxy with censored DNS server IP

IP Address:	192.168.100.254
Broadcast Address:	192.168.100.255
Subnet Mask:	255.255.255.0

Table 4.2: Internal network ("ProxyNet") configuration of the HonSSH proxy

The configuration of Table 4.1 indicates that IP 10.0.2.15 has been arbitrarily chosen in the VirtualBox standard NAT network 10.0.2.0. The *Default Route* is via IP 10.0.2.2, which represents the gateway of the network. Also, for external connections, the Domain Name System (DNS) server of the TU Dortmund University is specified as the *Primary DNS Server*. Thus, the SSH proxy can be reached by the attacker addressing the external IP of the host on port 22, and the SSH proxy can access the Internet via the NAT connection. This is necessary because the attacker is scanned by the SSH proxy as soon as he can successfully connect to the honeypot.
Inside the proxy system, the connection and communication are accepted, processed and forwarded in an internal network by HonSSH [98]. HonSSH is the software realization of the SSH proxy written in the programming language Python [111]. The destination of the forwarding, inside the SSH proxy, is the IP address (192.168.100.4) of the honeypot on port 22. The configuration of Table 4.2 does not show any peculiarities. The IP address and the network can be chosen as preferred.

The SSH proxy is directly connected with the honeypot (via an internal network "ProxyNet"). This honeypot is the destination of the attacker, after being forwarded from the host system via the SSH proxy. There are two virtual NICs on the honeypot; each configured for a separated internal network. One adapter for the "ProxyNet", as mentioned above, and the other adapter for another internal network, the "HoneyLabNet". The assumption applies again that exclusively VMs within the same internal network can communicate

with each other. The two virtual NICs are configured as follows:

IP Address:	192.168.100.4
Broadcast Address:	192.168.100.255
Subnet Mask:	255.255.255.0
Default Route:	192.168.100.254

Table 4.3: Internal network ("ProxyNet") configuration of the honeypot (DMZ)

IP Address:	192.168.10.112
Broadcast Address:	192.168.10.255
Subnet Mask:	255.255.255.0

Table 4.4: Internal network ("HoneyLabNet") configuration of the honeypot (DMZ)

The configurations of Table 4.3 and Table 4.4 have common settings for user-defined networks: Both networks and IP addresses are arbitrarily chosen. The only setting to consider is to set the *Default Route* to the SSH proxy. Therefore, the IP address of the SSH proxy is set as the gateway of the "ProxyNet". This will enable the honeypot to communicate through the transparent proxy. There is no need to explicitly define a gateway for the "HoneyLabNet" since it is implicitly set. For instance, in this case, it has the IP address 192.168.10.1.
Now, the attacker can reach further systems/honeypots in the "HoneyLabNet". For simplification, there is only one other honeypot, with Microsoft Windows, located in this internal network. The honeypot has just one virtual NIC which is configured as follows:

IP Address:	192.168.10.51
Broadcast Address:	192.168.10.255
Subnet Mask:	255.255.255.0
Default Route:	192.168.10.1
Primary DNS:	192.168.10.1

Table 4.5: Internal network ("HoneyLabNet") configuration of the honeypot (HL)

As mentioned above, a common configuration for the chosen network. This honeypot provides several open/filtered ports which are usually used on a Windows system. In addition, port 22 has been opened to offer an SSH service.

Recapitulating, the attacker is forwarded from the host to the SSH proxy. On the proxy system, the (encrypted) network connection is accessible in plain text and can be analyzed.

Afterward, the SSH proxy forwards the data via a second SSH connection to the first honeypot. The entire process is invisible to the attacker. From the attacker's point of view the SSH connection is directly established on the honeypot. This honeypot can be probed and tested. As a result, the attacker can find the internal network "HoneyLabNet" which contains another honeypot. In this isolated network (HL) he can use techniques against Windows systems and start a lateral movement.
The topology is simply designed and offers many capabilities to adjust and configure the internal networks as required, for instance, to neatly be incorporated in a corporate network.

4.2 Prepreparation of a Transparent Proxy with HonSSH

In order to have as much control as possible over the attacker and his activities, a transparent proxy is connected before the actual honeypot. Both systems in conjunction allow a wide range of possibilities to recognize the attacker. Whether by active measures of the proxy system or passive measures, which mainly affect the honeypot.
The foundation for the SSH proxy is an ordinary and up-to-date Linux system "Ubuntu 16.04 Desktop (64-bit)" [12]. This system is completely updated and was hardened against attacks by measures such as only providing the necessary services and using strong passwords.
The network connections are established as described in section 4.1 "Topology". The SSH proxy can access the Internet over a NAT and the honeypot via an internal network. The attacker is unconsciously forwarded from the SSH proxy via the internal network to the honeypot. More precise details about the network and routing are given in the section aforementioned.

Once the operating system is installed and the network connections are properly configured, the actual proxy can be installed. The publicly available software project HonSSH [98] is used for this purpose since it is permitted for both: commercial and private use. HonSSH is written in the programming language Python and is continuously developed further. All the particular requirements made on the SSH proxy are met. It also offers additional features to extend or adjust the approach of the thesis. As the source code is available, it is also possible to customize it to individual needs.
By default, HonSSH captures and saves any connection attempts, interactions and downloaded/copied files of the attacker. The captured data are arranged by session or general log records, such as connection attempts, by date. A weak password is selected as a vulnerability for the honeypot. HonSSH supports this kind of vulnerability without modification. This scenario can even be enhanced because HonSSH supports "Password-Spoofing": Forged credentials for individual users can be defined, and whenever an attacker submits

one of these they are replaced by the real password. Alternatively, a "random chance" can be specified with which the attacker will be successfully logged in, no matter what password he has entered. Both possibilities allow an attacker to log on to the honeypot with bogus credentials. However, these credentials cannot be used on the honeypot and do not offer any identification or recognition capabilities. Therefore, this feature is not used in the approach of the thesis. Another feature allows to view and even hijack a session of an attacker in real-time. While this is valuable for real use cases, it is negligible for the research since there is no necessity to interact with an active attacker personally. A required condition of the SSH proxy is to forward the attacker's IP as it is needed in the honeypot, for instance for journals. This is achieved by using the "Advanced Networking" feature of HonSSH which is a kind of IP spoofing. Furthermore, there are several hooks in the application which are triggered by events and can execute custom scripts. For example, there are events for establishing a connection, logging on to the system, or running a command.

HonSSH already has many valuable features and the installation and configuration are straightforward. The main configuration file of HonSSH is "honssh.cfg". In total, this file has 738 lines of text with many comments. In the following, a short extract is presented to introduce the structure and the first configuration parameters. Then, only the relevant segments are described.

Listing 4.1 describes the accessibility parameters (input) of HonSSH and the IP of the NIC which will realize the forwarding in the internal network "ProxyNet". The SSH proxy is reachable via NAT at IP address 10.0.2.15 (line 7) on port 2222 (line 14). In the network "ProxyNet" it is assigned the IP address 192.168.100.254 (line 21).
The configuration file has to be customized according to this schema. By default, most of the parameters are properly set. Nevertheless, the entire file and settings should carefully be read and verified to avoid a misconfiguration. For example, the IP address ([honeypot-static]: honey_ip = 192.168.100.4) and the port ([honeypot-static]: honey_port = 22) of the honeypot are essential. Furthermore, the "Advanced Networking" ([advNet]: enabled = true) and the capturing of downloaded and copied files "File Download" ([download]: passive = true and active = true) has to be enabled. Finally, a hook will be defined in the "Application Hooks" ([output-app_hooks]: enabled = true) to enable an automated scan of the attacker after a successful login on the honeypot (login_successful = /opt/honssh/roa/honssh_scan.py). The custom script "honssh_scan.py", located in the subdirectory "roa" of the HonSSH project, accomplishes this task.

```
1  [honeypot]
2
3  # IP addresses to listen for incoming SSH
       connections.
4  #
5  # input: IP Address
6  # required: YES
7  ssh_addr = 10.0.2.15
8
9  # Port to listen for incoming SSH connections.
10 #
11 # input: Number
12 # required: YES
13 # default: 2222
14 ssh_port = 2222
15
16 # IP addresses to send outgoing SSH connections.
17 # 0.0.0.0 for all interfaces
18 #
19 # input: IP Address
20 # required: YES
21 client_addr = 192.168.100.254
```

Listing 4.1: Extract of the configuration file honssh.cfg

Once started, the SSH proxy is ready and automatically listens for incoming connections. There is no need to mind technical configurations on the system such as internal forwarding or spoofing. It is also not necessary to run an SSH server on the proxy or to configure IP forwarding. HonSSH takes care of all this.

As already mentioned in section 3.3 “Recognition Measures”, a port and service scan of the attacker is unlikely to yield strong attribution features. However, it enhanced the gathered information about the attacker and can reveal parts of his infrastructure. The trigger for the automatic scan is a successful login. This event reduces the number of scans to the necessary minimum and filters ordinary noise such as incoming port scans.
The script initiates “On-Demand Scanning” of the attacker’s host by using the Application Programming Interface (API) of Shodan. The result contains an identifier on Shodan which is written into a log file. Unfortunately, Shodan does not provide results in multiple cases during the testing stage although the host was online. The delivered message

indicated that the target network was recently scanned. Attempting to fetch the information results in another message that no information is available for this host. Therefore, it is not enough to rely on Shodan and enhance the script by a Nmap [83] scan. This tool is an open-source security scanner which supports, for instance, network exploration, port scanning, service and OS discovery. Nmap was called with the arguments "-v -A -sV –version-all -PN". The results are written in the same log file that Shodan uses. This affects that the most widely used 1.000 TCP ports are scanned. The additional arguments also provide for an attempt to discover the corresponding services and operating system. Furthermore, host discovery is skipped since Nmap will not scan a host if it seems to be offline (not responding to ping probes).
Python libraries exist for Shodan and Nmap, therefore both scans are straightforward. An API key is only necessary for Shodan which can be obtained from the official website [85].

4.3 Prepreparation of a Linux High-Interaction Honeypot

The honeypot, the attacker uses as an entry point, mimics a test or maintenance system. It is necessary to use a High-Interaction Honeypot to gather as much specific data as possible regarding the attacker. Any other kind would limit the level of interaction and therefore the possibilities and techniques to gather information. This reduces the number of publicly available solutions. Moreover, all commercial solutions are not affordable for the financial constraint of the thesis. During the research, it became noticeable that Low/Medium-Interaction Honeypots or custom solutions are mostly used [56, 2]. Considering the primary task of a honeypot, this is reasonable. Except for scientific research, it is sufficient to detect an attack and to disallow further interactions. Also, no public available solutions for a High-Interaction Honeypot was able to meet the requirements. For instance, most attackers change a weak password to a strong password to prevent other attackers from compromising the system [2]. This requires a solution which allows an attacker to change the weak password to a strong password (strong attribution feature) and at the same time to keep the weak password to enable other attackers to compromise the honeypot. While this is a necessary condition for this thesis and evaluation, this requirement can be ignored, if this approach is used in a productive network such as an organization. It should be sufficient to allow an attacker to change the password because it is unlikely to deal with multiple autonomous attackers at the same time.

For this approach, a regular Linux distribution "Ubuntu 16.04 Desktop (64-bit)" [12] is modified to build a honeypot. This process is described in detail in this section. As a PoC the trade-off between realistic functionality and time consumption for preparing the honeypot was faced. Most implemented techniques for monitoring, deception and detection/identification were inspired by malware, rootkits or simple analysis tools. For

instance, userland rootkits used preloading of malicious shared libraries to inject code without manipulation the files itself. This is a well-known technique but should be sufficient for the PoC. A better solution is to hook the specific system calls or directly modify the code. However, this could be an improvement for future work, since this thesis is focused on recognition of attackers rather than building a bulletproof honeypot.

4.3.1 Basics

Besides the honeypot functionality, some basic preparations have been made. An SSH server was installed and configured. Listing 4.2 shows only the changed parameters: The logging level was increased to gather more information, "PasswordAuthentication" was enabled and several other authentication methods were disabled. Also, "X11Forwarding" was disabled since it is disallowed by the SSH proxy and the attacker should be offered a consistent image.

```
# Logging
LogLevel DEBUG

# Authentication:
RSAAuthentication no
PubkeyAuthentication no

# Change to no to disable tunnelled clear text passwords
PasswordAuthentication yes

X11Forwarding no
```

Listing 4.2: Extract of the parameterization of sshd_config

Furthermore, the central directory "/etc/roa" was created to store the gathered information. This directory is outside of the attacker's privileges and is protected and hidden by the measures described in the following. It contains important files such as the main logging file "roa.log", lists for the injectable credentials and URLs, and the gathered information from Sysdig and Falco.
A fake bash history, shown in Listing 4.3, is created to lead the attacker to find and extract the information which results in strong attribution features. Finally, the permission of the user account with the weak password is restricted, and a real root user with a complex password created. This limits the options for the attacker and forces him to reveal his skill set or leave the system.

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install openssh-server
sudo su
su root
ping 192.168.10.51
remmia
ssh smpatroc@192.168.10.51
exit
```

Listing 4.3: Fake .bash_history to lead the attacker

4.3.2 Monitoring

The most important functionality for a honeypot is complete monitoring. Everything an attacker does should be logged and controlled. There are several ways to provide this functionality. For example, system calls or references to shared libraries can be hooked, Linux system utilities can be modified or used, or external analysis/forensic tools can be applied. Depending on skills and resources such as time and finances, suitable options are available. For this thesis, time and finances are crucial. Hence, the external open-source tools Sysdig [24], Falco [23] and Filebeat [33] (of the "Elastic Stack") are used.
The "Elastic Stack" with Elasticsearch [32], Logstash [35], Kibana [34] and Beats [31] such as Filebeats is a powerful open-source solution for handling, searching and analyzing data. This software stack has to be installed on the "Logging System" which will collect and analyze the data. For the honeypot, it is only necessary to transfer the log files to a central "Logging System". In order to accomplish this, Filebeat is installed. The decision to choose these tools was made on the basis of their optimized interaction. Sysdig monitors the system state and any activity of a Linux system, and Falco completes this with behavior monitoring to detect the anomalous activity of applications and specific system calls. The output of the captured data is given to *syslog* [135] whose output can be loaded and analyzed by the "Elastic Stack" by default. As mentioned before, the data transfer to the "Logging System" is not used in the PoC, but can be customized according to the implemented workflow. It is only required to configure Filebeat, since its process and transactions are already respected in the implementation.
To monitor the system, the open-source monitoring tool Sysdig is used. Since Sysdig analyzes the system calls it is not affected by the implemented userland rootkit functionality. It will still monitor anything running on the honeypot and log it for the manual analysis performed separately and offline. The following list is not exhaustive, but indicates the main activities which are inspected:

- system calls
- raw network traffic
- network connections
- running processes
- opened files

This enables Sysdig to trace the system and applications, which is perfect for forensic and analysis tasks.
Beside of the supply of various Linux utility functions, Sysdig offers little scripts written in Lua [64] to analyze the captured event stream. These scripts are called "chisels" and perform useful analysis action such as finding slow system calls (*bottlenecks*), watching log files (*spy_syslog* and *spy_logs*) or monitor user activities (*spy_users*). All these possibilities and more are described in the substantial documentation [130] of Sysdig. There are decent guides for installation, configuration, and chisels.
After installing Sysdig and Falco, it is recommended to build a simple service which starts and stops Sysdig with desired flags. For this PoC Sysdig is run with the flag *-w* which writes the captured events into a file. This is useful for manual analysis. However, this could also be done with Filebeat, to transfer the data automatically in defined periods or even continuous in real-time. The start/stop script (*roa_sysdig*), located in */etc/init.d*, contains to methods: *do_start()* and *do_stop()*.

```
1 do_start()
2 {
3    sysdig -w /etc/roa/roa_image.$(date +"%Y%m%d-%H%M%S").gz
        > /dev/null 2>&1 &
4 }
5
6 do_stop()
7 {
8    killall sysdig
9 }
```

Listing 4.4: Simplified code for starting sysdig as a service

Both methods are straightforward. *do_start()* runs Sysdig and stores the captured data into */etc/roa/* (this folder is used to save all log files of the honeypot). The acronym "roa" comes from "recognition of attacker" and will be used as a magic string to identify several

modules, e.g., for hiding. Each file has the prefix “roa_image.” followed by a formatted string of the actual date and time and a suffix “.gz”. “> /dev/null 2>&1 &” runs the command in the background and redirects any possible output (standard I/O streams [127]) to */dev/null* (any data will be discarded). do_stop() simply kills the corresponding process [71].
Finally, the services should be started during the system startup by using the command *update-rc.d* [153]. Now, every activity on the honeypot is logged and can be analyzed.

4.3.3 File and Process Hiding

In this subsection, the implemented rootkit functionality is described. Rootkits use these and other techniques to hide their malicious activities, while the honeypot is equipped with it to obfuscate the monitoring, logging, and communication. The kernel or the user-level could be modified to achieve this. As mentioned before, the trade-off between realistic functionality and time consumption was faced. Therefore, the decision was made to modify the userland by manipulating various operating system functions.

Most of the hooks were implemented by injecting a shared library, written in the programming language C [116]. This technique is also used by Linux rootkits such as “Jynx2” [9]. This is a neat and straightforward solution on the one side, on the other, it is mostly trivial to detect and remove. However, it is possible to hinder or even counter several detection methods [60] with an appropriate overhead.
Other options would be to hook system calls or to replace the corresponding binaries with modified versions. Both solutions require much work to create a honeypot and therefore were rejected. For deployment in a productive network, this could be a worthful advancement, but it is too much overhead for a PoC. A similar procedure was used in subsection 4.3.4 “Authenticating Manipulation” because the preloading trick is infeasible if the SUID (Set User ID) or SGID (Set Group ID) bit is set. By setting this bit, the program runs with the permission of the caller and the owner. This means anybody with access to a script owned by root could manipulate it and run it with root privileges. Therefore, the authentication is directly modified and replaced.

To make the preloading approach more tangible, it will be introduced, and an example will be given for the system utility *ls* [82]. Shared libraries provide modules (functions) which can be accessed via an API. When a program that uses dynamic libraries is compiled, references (undefined symbols / corresponding API calls) to these functions are left without relation to a specific shared library. These references are resolved at runtime. A dynamic linker (such as *ld.so* [77]) loads the necessary shared libraries hierarchically and links the first occurrence to the process. Hence, the injected shared library should be loaded before

the regular shared libraries. To achieve this, the environment variable *LD_PRELOAD* or the file */etc/ld.so.preload* in the system search path can be used, because they are loaded before the shared libraries in */lib* and */usr/lib*. The result of the dynamic linking is shown for *ls* (unmodified in the honeypot) by using *ldd* [76]. The output is the following:

```
$ ldd /bin/ls
    linux-vdso.so.1 =>  (0x00007ffc00e6b000)
    libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so
        .1 (0x00007fdcaa987000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0
        x00007fdcaa5bd000)
    libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0
        x00007fdcaa34d000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0
        x00007fdcaa149000)
    /lib64/ld-linux-x86-64.so.2 (0x00007fdcaaba9000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so
        .0 (0x00007fdca9f2c000)
```

Listing 4.5: Loaded shared libraries for /bin/ls without injected shared library

As the example shows, various shared libraries are loaded by *ls*. The use of the preloading trick, also known as the concept of "function interposition", the injected shared library is loaded before any other library (shown in listing 4.7). The dynamic linker resolves the references by hierarchically checking the shared libraries. Since the injected shared library is loaded first, it is possible to hook any of these references. For *ls* this is *readdir()* [113] which is defined in the C standard library (for this honeypot *libc.so.6* [78]). To find these references, the source code of the utilities can be reviewed, or tools such as *sysdig* can be used to trace the system calls (not hooked in this PoC).
After identifying the module, it has to be hooked in the injected shared library. In the following the source code for the *readdir()* function is outlined:

```
1  #define _GNU_SOURCE
2  #include <dlfcn.h>
3  ...
4
5  void *libc = dlopen("/lib/x86_64-linux-gnu/libc.so.6",
       RTLD_LAZY);
6
7  static struct dirent *(*old_readdir) (DIR * dir);
8  struct dirent *readdir(DIR *dirp){
9      struct dirent *dir;
10     old_readdir = dlsym(libc, "readdir");
11     while((dir = old_readdir(dirp))){
12       if(strncmp("roa", dir->d_name, 3) == 0 ) break;
13     }
14     return dir;
15 }
```

Listing 4.6: Simplified and modified code from Jynx2 [9]

The code is straightforward: A handle for the shared library *libc.so.6* is loaded with a lazy binding (only resolved if the reference is called) and accessible via the variable *libc* [25]. Then a static pointer of a structure type for the original *readdir()* function is defined. This will be used to call the original version of *readdir()* to conserve the functionality. Therefore, it is not necessary to rewrite the whole code. In line 8 the modified version of the *readdir()* function is defined. In this function, the reference to the *readdir()* function is resolved dynamically to get the original function from *libc.so.6* [26]. Then a simple check is performed to find the substring "roa" in the output of the original *readdir()* [133]. If this is the case, the directory/file is skipped. Finally, the filtered output of the original *readdir()* is returned. As a result, all files or directories with the substring "roa" are hidden from *ls*.
Unfortunately, it is not sufficient to hook *ls*. Further system utilities from the non-exhaustive list below have to be hooked because they can reveal the monitoring and communication:

- *ps* [110]
- *find* [44]
- *top* [148]

Also, the added monitoring tool Sysdig and the log file shipper Filebeat has to be hidden.

To achieve this, most malicious rootkits hook various modules. One advantage is that they have more control and can manipulate and inject arbitrary functions at any situation such as opening a file or write into a file. For this PoC it is sufficient to hide information and therefore the *readdir()* hook is extended according to the same principle as mentioned before. For instance, processes are accessed via */proc/PID*, where PID is a subfolder named by the process ID of every running process on the system. To hide a process the subfolder of that process has to be skipped, when *readdir()* access these data. By getting the process name for a process ID, the idea is analogous to the example given above.
After hooking all needed utilities or modules, the code can be compiled into a shared library with position-independent code (option *-fPIC*) and link it with "Dynamically Loaded" libraries (option *-ldl*). Finally, the path can be added into the */etc/ld.so.preload* to enable a "global" preloading. For the PoC the shared library is called *libr.so* and copied to one of the standard library locations */lib/x86_64-linux-gnu/*.
The output from *ldd* for a hooked version of ls with the injected shared library *libr.so* could be as follows:

```
$ ldd /bin/ls
        linux-vdso.so.1 =>  (0x00007ffc0fdfe000)
        /lib/x86_64-linux-gnu/libr.so (0x00007f41d5c68000)
        libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux
            .so.1 (0x00007f41d5a46000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0
            x00007f41d567c000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0
            x00007f41d5478000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3
            (0x00007f41d5208000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f41d5e6b000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread
            .so.0 (0x00007f41d4feb000)
```

Listing 4.7: Loaded shared libraries for /bin/ls with injected shared library

4.3.4 Authenticating Manipulation

Complex passwords are strong attribution features. The honeypot uses a weak password as a vulnerability because attackers often change weak passwords to strong passwords after compromising a system [2]. This fact could be used as a strong recognition feature because

no uninvolved attacker should know this strong password. Therefore, it is necessary to allow attackers to change the password, and at the same time to keep the weak password for other potential attackers. To evaluate the approach, different independent attackers should be analyzed to find correlations and separations.
If this approach is used in a productive environment, the chosen password should be easy to guess but not be found in common password lists [92, 61]. For instance, the company name with the actual year as a suffix can be used since this password is easy to guess for an attacker who targets the organization.

Pluggable Authentication Modules (PAM) [103] is a centralized authentication mechanism used by Linux systems. It offers a flexible and modular architecture which can easily be used for various applications. The PAM library is the core of the PAM architecture. Applications which use functions of the PAM module are directly linked to it and can be accessed through API calls. For every authentication process, the program calls these functions. The PAM library has four modules: *auth*, *account*, *password*, and *session*. Each of these modules offers basic functionality corresponding management tasks. Of particular interest is the module *auth* with the "Shared Object" (compiled library file) *pam_unix.so*, which contains the default authentication processes. Hence, it is ideal for implementing appropriate manipulations to enable multi-password support for user accounts.

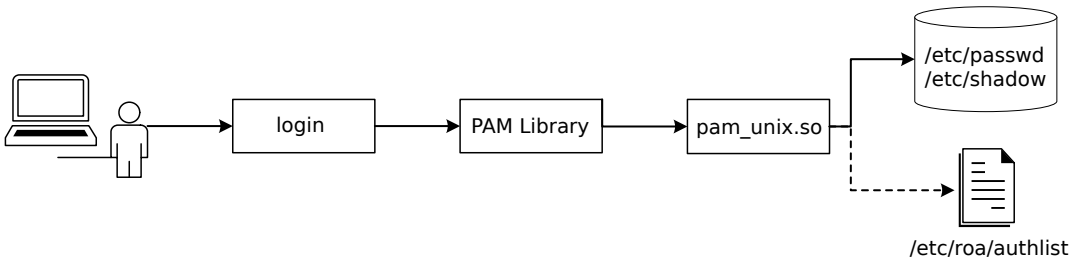

Figure 4.3: Outline of the user authentication in Linux

Figure 4.3 illustrates the use case of user authentication. The user calls the program *login* [81] by entering his username and password. This program calls the method *pam_authenticate()* [104] from the PAM library's API. In the library, the data is passed to the file *pam_unix.so* which verifies the username and password against the files */etc/passwd* and */etc/shadow*. The solid lines describe this standard workflow. In the shared library *pam_unix.so* the verification process is done in the function *pam_sm_authenticate()*. The dashed line shows that there is a modification which also verifies the username and password combination with */etc/roa/authlist* which is a simple text file with usernames and passwords. For authentication, it makes no difference whether the user is registered in the common password files or the added password list.

This modification has the advantage that any program which needs a user authentication verifies the username and password with the injected list of credentials. For instance, *su* [134] and *sshd* [126] are also affected by this modification.
To allow the attacker to change the password and at the same time keep the old password another modification was performed. If the method *pam_chauthtok()* [105] from the PAM library is called for example by the program *passwd* [107], the shared library *pam_unix.so* activates the method *pam_sm_chauthtok()*. The user has to enter the old and a new password. Then the new password is validated by a configured password policy such as length and complexity. If all checks are passed the new password is hashed and written into the */etc/shadow* file. The modification is simple: Instead of calling the hash function with the new password, the function is called with the old password. Additionally, the username and new password are appended to the password list */etc/roa/authlist*. This allows the attacker to "change" the weak password into a strong password and to keep the weak password for further attackers.
Besides *passwd*, also *adduser* [1] and *useradd* [154] are affected which also allows the attacker to add new users to the system.

From a more technical point of view, the files *pam_unix_auth.c* and *pam_unix_passwd.c* are modified. Then the PAM library must be recompiled and the shared library *pam_unix.so* must be replaced. Finally, the timestamp of the replaced *pam_unix.so* must be changed to obfuscate the modification. This is achievable with the *touch* [149] command and an option like *-r* (*–reference=FILE*). For the honeypot used in this PoC, the file is located in */lib/x86_64-linux-gnu/security/*.
The Code itself is merely read and write a file with a comparison of strings. Furthermore, the events are logged into */etc/roa/roa.log*.

The injection point in the *pam_unix_auth.c* is after:

```
/* verify the password of this user */
retval = _unix_verify_password(pamh, name, p, ctrl);
```

Listing 4.8: Code fragment of pam_unix_auth.c

And the injection point in the *pam_unix_passwd.c* is after:

```
/*
* use_authtok is to force the use of a previously entered
* password -- needed for pluggable password strength
    checking
*/
retval = _unix_read_password(pamh, lctrl
                             ,NULL
                             ,_("Enter new UNIX password: ")
                             ,_("Retype new UNIX password: ")
                             ,_UNIX_NEW_AUTHTOK
                             ,&pass_new);
```

Listing 4.9: Code fragment of pam_unix_passwd.c

Furthermore, the line

```
tpass = create_password_hash(pamh, pass_new, ctrl, rounds);
```

Listing 4.10: Original line of code from pam_unix_passwd.c

is changed into

```
tpass = create_password_hash(pamh, pass_old, ctrl, rounds);
```

Listing 4.11: Modified line of code from pam_unix_passwd.c

4.3.5 Injection of Unique Knowledge

Strong attribution features are necessary to recognize an attacker. For example, changing a weak password to a complex password allows identifying an attacker since it is highly unlikely that autonomous attackers generate or use the same complex password. The changing and usage of such a strong password indicates a relation between the attackers and in this thesis, they will be seen as the same attacker, even if it is a group with shared knowledge. The implementation of this process is described in subsection 4.3.4 “Authenticating Manipulation”. In order to avoid having to rely exclusively on this feature or serious mistakes of the attacker, further strong attribution features are injected into

every session. The attacker can extract these features in the form of information with some effort, and gains compromised knowledge. If he uses this unique knowledge over several sessions, it is possible to establish a clear correlation between these sessions.
The passive measures, mentioned in section 3.3 “Recognition Measures”, are deployed on the honeypot. Firstly, unique credentials are injected into every session, and secondly, a unique URL (more accurate the path of the URL) is placed in every session. Both injections are realized by using a custom Python script which is executed at every new SSH session. The script is named “roa_injector.py” and is located in the directory “/etc/roa/”. To run this script at every SSH session the file “/etc/ssh/sshrc” was created with the content “python /etc/roa/roa_injector.py”. Finally, the file “/etc/ssh/sshrc” was made executable. Besides the injection of unique information in files, the script also resets the timestamp of the manipulated files to the timestamp before the manipulation. Furthermore, it writes all the injected unique information and parameter of the session, such as username, password, and timestamp, into the log file “/etc/roa/roa.log”. These log entries are beneficial to compare the information and straightforwardly find links between the sessions.

Injection of Unique Credentials

The injection of unique credentials has to be done in such a way that the attacker can build up knowledge and cannot just reference these credentials. For example, a simple script which establishes a remote connection is unsuitable. Also, bogus entries in the bash history are too easy to pick out, and it is unfeasible to exchange a plain text in every session unnoticeably. Therefore, the information must be placed in a manner that it can be found, require some effort to extract it and is not considered as a decoy. The effort which is necessary to extract the information can additionally be used to challenge the attackers and categorize them. On a targeted attack, it can be assumed that the attacker has particular capabilities; however, in the evaluation of this thesis with random attackers, the assumption is not necessarily accurate. In this case, the extraction of the injected credentials also serves as a filter mechanism which will sort out bots and attackers with tool-limited or no skill set. These kinds of attackers have to be detected, but they are neither able to find and extract the information nor will they invest enough time to build up a solid knowledge.
A valid scenario for injected credentials is a remote connection to another system. This will be the entry point into the HL and a direct connection to the next honeypot. Since the attacker should have as many different resources and challenges as possible to probe, and Windows systems are primarily use in corporate networks, the second reachable honeypot is a Windows system. The first honeypot is a Linux system; hence, software is required to establish a remote connection from a Linux to a Windows system. Remmina [50] is

such a software. By default, this remote desktop client is pre-installed on the used Linux system of the honeypot (Ubuntu 16.04), and it supports many network protocols such as Remote Desktop Protocol (RDP) and SSH. Moreover, Remmina allows injecting unique credential which can be extracted by an attacker with sufficient effort. Technically, it is not possible for an attacker to run Remmina directly to establish a remote connection to the Windows honeypot. He has to extract the credentials manually. To lead the attacker to Remmina, false entries in the bash history has been made to forge a launching of the software, and in addition, desktop shortcuts have been placed. Another entry in the bash history pretended an SSH connection to the Windows honeypot. If the attacker combines this information and slightly researches for Remmina, he will find that Remmina encrypts stored credentials, but in the standard configuration, they can easily be decrypted with little effort. Discussions and even source code are publicly available to decrypt the data [79].

Remmina creates a hidden folder in the user's home directory (~/.remmina). For each saved remote connection, a corresponding configuration file is located in this folder. Besides these, there is a general configuration file for Remmina itself. The use case of the thesis includes two files since there is only one remote connection to the HL. Remmina's configuration file is named as "remmina.pref" and contains general parameter such as resolutions, main_width and main_height and a parameter called *secret*. The other file contains parameters for the remote connection. The name has a 13 digit prefix which is a Unix timestamp and a suffix ".remmina". For instance, the file name used in the thesis is "1528704394644.remmina". This configuration file includes, among other parameters, the encrypted password and the username in plain text.

The encryption is a Triple Data Encryption Standard (3DES) [70] in Cipher Block Chaining (CBC) mode [30]. It is a symmetric encryption so that the same key (password) is used for encryption and decryption. Furthermore, it is a block cipher with 64-bit blocks. This indicates that the plaintext has to be a multiple of 64 bits. If the plaintext is shorter than the required length, it will be padded by adding additional bytes at the end of the block.
"remmina.pref" contains a base64 [68] encoded *secret*. Decoded it has a length of 32 bytes which meets the CBC criteria. The first 24 bytes represents the key for 3DES, and the last 8 bytes are used as an Initialization Vector (IV) for the CBC mode.
Decryption works as follows: First, the encrypted password (ciphertext) is base64 encoded and has to be decoded. The result has a length of a multiple of 64 bits (in the PoC 8 or 16 bytes) which fits the required block length. Then, the decryption function for 3DES is called with the IV and the key, which are encoded in *secret*, and the ciphertext (password) from the remote connection configuration file.

Encryption works almost precisely the same, only with operations in reverse order: First, the length of the password (plain text) has to be a multiple of 64 bits. If it is too short, it will be padded by adding null bytes at the end (null byte in Python b“\0”). Then, the encryption function for 3DES is called with the IV and the key, from *secret*, and the padded plaintext. Finally, the ciphertext will be encoded with base64.

In the PoC, anything in the configuration files except the username and the password can be considered as a constant. Hence, the process described above can easily be automated to inject the credentials with an encrypted password. It takes enough effort to extract the password that the attacker will memorize it. To increase the likelihood of him doing this only weak and easy to remember passwords, such as “summer2018”, “football”, “password1” or “123456”, were chosen. After the first remote connection, the username is stored in the bash history. This is beneficial since the username in the configuration file is stored in plain text. Of course, the list of credentials is finite; however, in the PoC a list with 25 unique entries is used, and it should be enough for a real scenario. If the attacker's knowledge is not compromised in 25 sessions, it is unlikely to be successful later on. Table A.1 in the appendix shows the list of credentials.
The Attacker can establish an SSH connection to the Windows honeypot by using the extracted credentials over several sessions. Since the credentials are unique and never reused, they are classified as strong attribution features.

Injection of Unique URLs

Another strong identification feature is the unique URL which is injected in every session. While in most cases the domain is invariable, the path has to be unique. It is essential that the URL sounds interesting and does not contain characteristics that stir up suspicion in the attacker. For example, conspicuous parameters or cryptic paths are to be avoided. Table A.2 in the appendix shows the unique paths which were injected in the PoC. Well-known paths or fragments are used to pretend sensitive data or potentially vulnerable or valuable web applications.
It has been slightly difficult to find a suitable injection point for the URL. The bash history is too natural to inspect and would reveal the manipulation. Bookmarks or a browser history are impractical since the attacker cannot use a graphical interface. As a solution for the PoC, a desktop shortcut is placed and modified every session by exchanging the unique URLs and resetting the corresponding timestamp. Unfortunately, the URL is stored in plain text. However, the attacker cannot access the Internet and navigate to the URL from the honeypot. Thus, it is suggested that he extracts the URL one single

time and visit it from his systems. Listing 4.12 shows the simple structure of the desktop shortcut.

```
[Desktop Entry]
Encoding=UTF-8
Name=NewWebsite
Type=Link
URL=<Unique URL>
Icon=text-html
```

Listing 4.12: Content of the injected desktop shortcut "NewWebsite.desktop"

A universal redirect was configured in the ".htaccess" file, to avoid programming a website for each injected URL. This text-based file is used to configure compatible web servers. Any request to the target domain with an arbitrary path (except the path of the redirect destination itself to avoid infinite loops) is redirected to the "maintenance.php" file. The technical realization of the redirection is shown in Listing 4.13.

```
RewriteEngine on
RewriteCond %{REQUEST_URI} !/maintenance.php$
RewriteRule $ /maintenance.php [R,L]
```

Listing 4.13: Content of the configuration file .htaccess

The whole process is as follows: As soon as the attacker navigates to the injected URL, he is redirected to "maintenance.php". The web server logs information such as the timestamp, the IP and the original requested (the injected) URL of this visit. Additionally, "maintenance.php" performs a simplified browser fingerprint which can be enhanced without constraints. In the PoC exclusively HTTP header fields such as the client IP and port, the hostname, the referrer, and the browser agent are analyzed. The attacker gets a simple website with a brief notification that the internal services are shut down for maintenance work, and the website is accessible in a short period of time. The ostensible estimated time is calculated by rounding up the current time to the nearest half hour.
As mentioned before, there are more advanced options. These require more time and some of them legal cover. For instance, a custom version of the JavaScript library jQuery [141] can be loaded which is extended by own appropriate functions. The attacker will see that a local version of an (up-to-date) version of jQuery is loaded which is not unusual for today's websites. With a sophisticated implementation of the own functionality and the need of JavaScript to use interesting functions of the website, it is feasible to execute arbitrary JavaScript in the browser of the attacker. Section 3.3 "Recognition Measures" shows further methods of browser fingerprints.

Since this thesis is focused on recognition of an attacker, it is sufficient to inject the unique URL and use a simplified browser fingerprint as a PoC. This should be enough to demonstrate that the knowledge of the attacker was successfully compromised. However, the advanced measures are interesting and offer many options such as an entire (vulnerable) CMS as a kind of web version of the HL or even a kind of "hack-back" by using JavaScript. The more exciting and interactive the web application, the more likely the mistakes of the attacker to reveal sensitive data or at least parts of his infrastructure.

4.3.6 Summary

The honeypot contains many concepts that are sufficient for a PoC but can be extended. It is accessible via SSH. The required monitoring has been implemented, but data transfer has not been automated. However, this does not interfere with the evaluation, since no contributing factors are violated. Furthermore, it is possible to change the initial weak password to a strong password which can be used for recognition. Further strong identification features such as unique credentials and a unique URL are injected in every session. Also, basic obfuscation techniques are implemented to hide the monitoring and logging.
Altogether, this is a simple High-Interaction Honeypot which meets the requirements for implementing further attribution capabilities.

4.4 Prepreparation of a simple Windows High-Interaction Honeypot

Representatively for the HL, a Microsoft Windows 10 Professional honeypot is deployed. This Windows honeypot is a target which can be found in many corporate networks. An entire network would provide even more options such as "Lateral Movement" inside the HL. This would enable to observe an attacker while performing his cyber kill chain (for example Reconnaissance, Intrusion, Exploitation, Privilege Escalation, Lateral Movement, and Exfiltration) in a bogus corporate network. Also, the honeypots could communicate with each other since any system in the HL is exclusively accessible via a honeypot. A deterministic and supervised communication can be distinguished from the attacker's interactions. However, for the PoC it is sufficient to deploy one honeypot in the HL since the transition is based and prerequisites a strong identification feature.

The default configurations of the installation procedure were chosen for the Windows honeypot. Before starting the evaluation, the honeypot was completely updated. During the evaluation no updates were installed to ensure comparability between the testers and

attackers. This should be reasonable since no highly critical vulnerability with a publicly known exploit was published in this period. After installing and updating the system, 25 unprivileged users with weak passwords were created. These are the credentials which will be injected in the Linux honeypot as strong identification features. All selected passwords, such as “summer2018”, “iloveyou” or “passw0rd”, are simple to remember and most of them can be found in publicly known password lists [92]. The administrator (“smpatroc”) was the only one who got a strong randomly generated password (“TWTsX=dPxq5+”).
The Microsoft RDP service was enabled and configured to present the attacker a complete picture. Therefore, all unprivileged users were allowed to use RDP, and the security restriction of clients without “Network Level Authentication” was disabled. RDP is not exploitable for the attacker in the PoC but it can be identified and probed and, furthermore, it enables a simple enhancement of the PoC to open up to further scenarios.
By default, an SSH server is not installed in Microsoft Windows 10 Professional. However, it is simple to install using standard Windows tools. Optional features such as an SSH server can automatically be installed via “Apps & Features”. For the honeypot, the default configuration is sufficient. It is only necessary to enable an automatic start-up for the service.
Finally, the Windows firewall has to be disabled in order not to restrict the attacker. Also, Windows PowerShell [90] and further standard administration tools for Windows are left enabled and unrestricted for the attacker. This should allow him to load and use all desired tools or reveal his tactics such as “living off the land” where he uses as much as possible tools from the victim’s system. Since the attacker only has an unprivileged user, he is limited by default and has to demonstrate his skills, for instance by performing a “Privilege Escalation”.

The Windows honeypot does not contain further measures to inject strong identification features. It is sufficient for the PoC to provide the transition and analyze the communication and cyber kill chain including the toolchain. However, the full potential is not yet exhausted.

Chapter 5

Evaluation

This chapter presents the results of the evaluation regarding the implemented recognition measures to prove the hypothesis:

> *A recurrent attacker cannot only be detected but recognized and tracked through multiple sessions by using appropriate measures based on honeypots.*

Based on the evaluation of these measures and their efficiency, the hypothesis should be verified or falsified. The overall result and also the particular measures can be assessed. Since a specific attacker model (minimum level of skills and expenditure of time) is required which cannot be guaranteed to be found in this short evaluation period, the evaluation is divided into two parts: Firstly, the measures in respect to the hypothesis are tested with "known attackers", and secondly, the results are validated by unknown/random attackers. The group of "known attackers" is primarily used to evaluate the hypothesis and introduced measures. It can be assumed that this group will devote the required time and has the capabilities necessary to move within the provided architecture. Their methods and procedures probably differ from real-world attackers; therefore, their results will be contrasted with the results of the group of random attackers. This proceeding is based on the fact that there is high uncertainty regarding unknown attackers: Neither the required capabilities nor the time and effort necessary to build up a compromised knowledge can be assumed.
It should be feasible to correlate the results of both groups although their motives and procedures differ. It is not relevant how or why they achieve to extract the information and gain the required knowledge since the attacker model only requires capable attackers. A more inconspicuous or overcautious procedure is expected from the "known attackers"; however, some capable random attackers may also behave similarly. Especially targeted attacks or APT groups will be as quiet and inconspicuous as possible, even if these are not expected to be part of the random attackers in the evaluation period. Their TTP

will be more advanced (e.g., Exfiltration) but in general, can be reduced to a similar level concerning the hypothesis.

5.1 Evaluation of the Known Attackers

The group of known attackers consists of only four participants since the requirements of capabilities and expenditure of time were relatively high for the short evaluation period. One of the testers is a computer scientist without a security background, but with a profound knowledge of Linux and operating systems including their construction and implementation. All other testers are security professionals (penetration testers and security engineers) with professional experience.
For this group, the research architecture is transformed into a Capture the Flag (CTF) in which flags instead of sensitive data can be found. In total, there are the following five flags:

In the source code of the injected website:	
<!– ZmxhZ3tpX2tub3dfYmFzZTY0fQ== –>	flag{i_know_base64}
In the Linux honeypot:	
/home/patrick-eval/.flag	flag{this_was_easy}
In the Windows honeypot:	
C:\Users\<injected username>\Documents\flag.txt	flag{i_am_inside}
C:\Users\Public\Public Dokuments\flag.txt	flag{i_found_a_share}
C:\Users\smpatroc\Desktop\flag.txt	flag{i_can_see_light}

Basically, all flags are irrelevant but should induce the testers to explore, test and probe the honeypots and networks to find and extract information like a real attacker. The flag with the content “i_can_see_light” is located on the desktop of the administrator account with a randomly generated complex password. No exploit was publicly known to escalate the privileges or to somehow get the flag at the time of creating the CTF. This flag was for testing only, to indicate if there was a severe flaw in the architecture.
All tester got the following scenario apart from this, however, no further information or restrictions:

> You have the role of an attacker who is targeting a company. Your objective is to compromise the corporate network, establish persistent access and exfiltrate sensitive information as stealthily and as long as possible.
>
> Due to the simplified test environment, it is preferable to phase the attack in reasonably separated sessions.

> The sensitive information is hidden in several flags. Pattern: flag{a_sensitive_ information}
>
> The target IP is: 129.217.xxx.xxx
>
> You also managed to enumerate a username of the target: patrick-eval
>
> Please put yourself in the scenario and operate weighing. The resources are the property of the TU Dortmund University and come partly from private collections (resources outside the target network).
>
> Thank you for your participation and good luck!

5.1.1 Exemplary Analysis of a Known Attacker

One attack of the known attackers (in this subsection only referred to as attackers) is analyzed as an example. The first penetration tester who tested the research setup is chosen to prevent any possible prior knowledge. This analysis is focused on the recognition measures in the context of the attack.

Gaining Access

The attacker used an automated tool to crack the weak SSH password. Since the username was known from the scenario, only password lists were required. Such attack tools perform a brute force attack by simply processing the password lists and reporting back whether a connection attempt was successful or not.
The attacker started his attack at about 18:15 and was successful at about 20:20. Several password lists were used (distinguishable due to the breaks and kinds of passwords). It was launched with a custom password list including permutations of $<season + year>$ (e.g., "Summer2018") and permutations of the username. However, publicly known password lists [92] were mainly used. In total, 1.374 connection attempts were seen in those two hours. This indicates that he used small lists and had stopped for a while.

Exploring and Attacking the Systems

The first login on the Linux honeypot was at approximately 20:21 and the last logout was at about 23:23. This signifies that the attack lasted three hours. Within this period of time, the attacker was able to gain access to all honeypots and read sensitive data (flags). Altogether, the attack was divided into ten separate sessions which, in some instances, were running in parallel. The majority of these sessions are irrelevant for the study since they were exclusively utilized to download further tools. The primary activities are spread over two sessions which will be focused on the analysis.
Table 5.1 gives an overview of the individual sessions with start and end times and the performed activities. The first session was initialized by the automated tool which cracked

the SSH password – no operations were performed on the Linux honeypot. Session 2 and 9 (*Attack Session 1 & 2*) were used for the actual attacks. These two sessions contain all relevant activities such as information gathering, searching for vulnerabilities, attempting to exploit the system or escalate privileges, and lateral movement. Five sessions (*Helper Sessions*) were used to download further tools, for instance, several PowerShell scripts and scripts for exploiting Linux systems. Another session was opened to read out running processes, configuration files (e.g., IP and SSH configuration) and files of the user management (/etc/shadow and /etc/passwd). Furthermore, one session was utilized for data exfiltration where sensitive configuration files of Remmina were uploaded to the attacker system.

No.	Start	End	Main Activities
1	20:20:57	20:20:57	Login (Automatic Tool)
2	20:21:20	22:30:53	Attack Session 1
3	20:31:04	20:31:07	Download: Linux_Exploit_Suggester.pl
4	20:33:26	20:33:29	Download: LinEnum.sh
5	20:37:27	21:00:44	Exploration: Processes, Network Configuration, etc.
6	21:04:43	21:04:48	Download: RPDscan.py
7	21:07:21	21:07:24	Data Exfiltration: Remmina Files
8	21:39:45	21:40:07	Download: power.zip (Multiple PowerShell Scripts)
9	21:58:19	23:23:19	Attack Session 2
10	23:12:23	23:12:26	Download: linprivchecker.py

Table 5.1: Overview of the attacker's sessions

Unfortunately, the attacker did not change the weak password of the Linux honeypot; therefore, no strong recognition features are available for the *Helper Sessions* 3, 4, 6, 8, and 10. However, all downloaded files (*Helper Sessions*) are used in Session 2 (*Attack Session 1*), suggesting a relationship of these sessions. *Attack Session 2* (Session 9) also has these downloads as a linkage since all files from the *Helper Sessions* are deleted. It seems that the attacker wanted to erase any traces even though the bash history and other log files were not cleared. Only Session 5 cannot be linked with other sessions since the increased knowledge of the system configuration cannot be traced in other sessions. Particularly interesting is the inconspicuous Session 7 in which the attacker exfiltrated the Remmina files. The unique credentials which were injected in Session 7 were extracted and reused in both *Attack Sessions 1 & 2*. Consequently, these three sessions (Session 2, 7 and 9) are linked by a strong recognition feature. Figure 5.1 outlines the relationship of all relevant sessions. The strong recognition features (unique knowledge) are highlighted in red and link Session 2, 7 and 9. A weaker linkage (dotted line) is between the *Attack*

Sessions 1 & 2 and the *Helper Sessions* since procurement, usage, and deletion can be traced over these sessions. Hence, all tools are publicly available the linkage is valid but not irrefutably certain. Under this laboratory conditions external factors can be eliminated, however, in general, these publicly available tools can be dropped from a third party (within a *Helper Session*), and the attacker can adapt them in his attack.

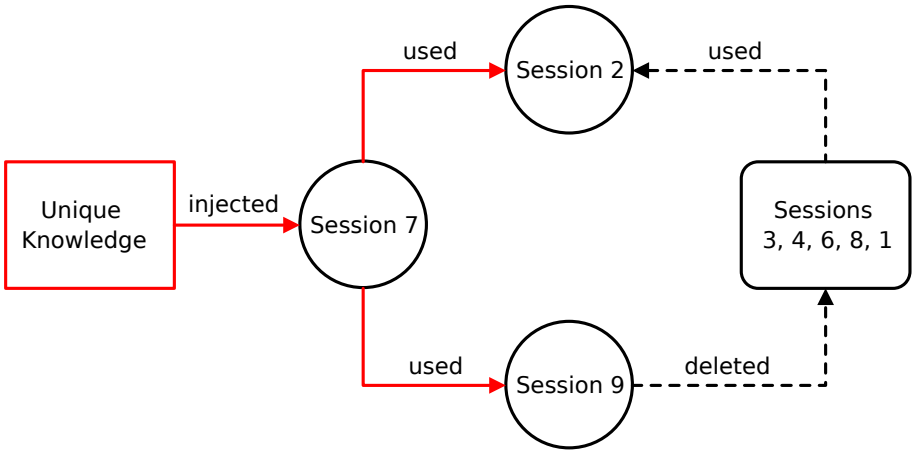

Figure 5.1: Transitions and relations of sessions

Except for the injected unique credentials no further strong recognition features were included in the attacker's knowledge since he has not paid any attention to the injected URL. Thus, it was not possible to get a browser fingerprint. Also, the automatic scan did not provide a valuable client fingerprint for recognition or attribution.

Generally, the actions of the attacker were precise and goal-orientated. After exploring the Linux system, he searched for vulnerabilities (tool based), further user accounts (enumeration through /etc/passwd and testing of common/standard user accounts such as "syslog", "admin" and "root") and potentially accessible systems (via Address Resolution Protocol (ARP) entries, iptables rules and routes). When Remmina was found, it was immediately inspected, and the injected credentials were extracted. With these unique credentials, a connection to the Windows honeypot was established. On the new system, he started from the beginning by searching for vulnerabilities and attempting to exploit the system (both tool based) automatically. Since no vulnerabilities or misconfigurations could be found on any of these systems, it was not feasible to exploit them, for example, to escalate the privileges. Finally, all downloaded tools were deleted.
It was striking that the attacker checked many configuration files for information gathering. Furthermore, he attempted to reconfigure the network connection since it was not possible to reach the Internet directly. Also, he used as much as possible from the given environment such as interpreters for his variety of different scripts (e.g., Python, PowerShell).

Summary

From the first login at 20:20:57 the attack lasted about three hours. However, the injected credentials were already used at 21:17:12, so after less than one hour after launching the attack.
The attacker did not change the weak password of the Linux system and did not pay attention to the injected URL. Therefore, no browser fingerprinting or further (active) measures by using the website were feasible. Shodan and Nmap yielded scan results, but the client fingerprint was insufficient. Though, it was possible to compromise the attacker's knowledge by injecting unique credentials, therefore all relevant sessions (*Attack Sessions 1 & 2*) can be linked reliably. Even the *Helper Sessions* have valid relations with a degree of uncertainty. The attack was performed within the At*tack Sessions 1 & 2*, and these sessions can be linked unambiguously. Consequently, it is feasible to recognize the attacker since the reused unique knowledge can be traced in arbitrary sessions even with multiple attackers.

5.1.2 Summary of the Results

Table 5.2 gives a summary of the recognition features for each attacker. An "X" signifies that the feature is present in the session while a "-" indicates absence. "o" is used for a particular case where a feature is present but there is no informative value. The first three attributes are strong recognition features; the last three are weak but valid features for recognition.

	Attacker 1	**Attacker 2**	**Attacker 3**	**Attacker 4**
Changed Password	-	-	-	X
Reuse Unique Credentials	X	X	X	-
Reuse Unique URL	-	X	X	X
Client Fingerprint	o	X	o	X
Browser Fingerprint	-	X	X	X
Downloaded Tools	X	X	X	-

Table 5.2: Overview of recognition features for each attacker

The table shows that each attacker has included at least one strong identification feature in his knowledge which therefore leads to a successful recognition. Furthermore, it is evident that just one attacker changed the weak password. This is regrettable and has to be reviewed with random attackers since the changing of a weak password to a complex password will immediately create a strong relationship between any sessions which contains this complex password. On average, the other feature could be integrated successfully in

the knowledge of the attackers and yield good results. However, client fingerprints via automatic scans should be revised due to the poor results. Often anything relevant is filtered by firewalls or routers.
It can be noted that all relevant sessions can be reliably assigned to a specific attacker. Most of the other sessions are insignificant since merely system properties (e.g., configuration files, running processes) were read or no critical operation was performed.
It is also apparent that the duration of the attacks is proportional to the capabilities of the attackers (penetration testers, security engineer, computer scientist). Moreover, it is interesting that all testers needed about one hour to extract and learn the first strong recognition feature.

In subsection 5.1.1 an attack of a known attacker was exemplarily analyzed with regard to recognition attributes. Such an analysis has to be performed for each attacker. However, since the proceeding is analogous, just an overview will be given.
The following is a schematic presentation of the analyzed attacks. The structure is orientated on the prior example. The chronological parameters are shown, and the individual sessions are broken down. Finally, it is reviewed which recognition features are present in the sessions and how reliable the relation between particular sessions is.

Attacker 1

First Login: 20:20:57 Last Logout: 23:23:19 Duration Attack: ~ 3 hours

Sessions: 10

1	20:20:57	20:20:57	Login (Tool)
2	20:21:20	22:30:53	Attack Session 1
3	20:31:04	20:31:07	Download: Linux_Exploit_Suggester.pl
4	20:33:26	20:33:29	Download: LinEnum.sh
5	20:37:27	21:00:44	Exploration: Processes, Network Configuration, etc.
6	21:04:43	21:04:48	Download: RPDscan.py
7	21:07:21	21:07:24	Data Exfiltration: Remmina Files
8	21:39:45	21:40:07	Download: power.zip (Multiple PowerShell Scripts)
9	21:58:19	23:23:19	Attack Session 2
10	23:12:23	23:12:26	Download: linprivchecker.py

Weak Recognition Measures:

Client Fingerprint: Insufficient

Browser Fingerprint: Not Available

Strong Recognition Measures:

Changed Linux Password: False

Reuse of Unique Credentials: True

Navigation to Unique URL: False

Further Recognition Measures:

Usage of Downloaded Tools: Weak Recognition (Publicly Available)

Reliability of Recognition for each Session:

Confident: 2, 7, 9

Plausible: 3, 4, 6, 8, 10

Not Feasible: 1, 5

Attacker 2

First Login: 13:32:23 Last Logout: 17:07:07 Duration Attack: ~ 3.5 hours

Sessions: 20

1	13:32:23	13:32:54	Password Guess + Logout
2	13:33:01	14:41:48	Attack Session: Exploration
3	13:51:26	13:56:24	Lateral Movement Attempt: Remmina and SSH
4	13:56:30	14:41:42	Attack Session: Lateral Movement
5	14:04:59	14:08:27	Check: Remmina
6	14:05:03	14:05:03	SFTP Attempt
7	14:20:43	14:20:45	Download: remmina_password_exposer.py
8	14:34:57	14:35:09	Data Exfiltration: Remmina files
9	14:42:23	15:15:45	Attack Session: Windows System 1/4
10	14:47:21	16:18:51	Building Infrastructure: SimpleHTTPServer
11	14:54:47	14:54:54	Download: Invoke-Mimikatz.ps1
12	15:15:55	15:27:37	Attack Session: Windows System 2/4
13	15:27:42	17:07:07	Attack Session: Windows System 3/4
14	15:52:20	15:52:22	SCP Command: scp -t ~
15	15:54:12	15:54:14	SCP Command: scp -t /tmp/
16	15:57:09	15:57:11	SCP Command: scp -t /tmp/
17	16:01:08	16:36:47	Attack Session: Windows System 4/4
18	16:06:55	16:07:40	Download: laZagne_x86.exe
19	16:19:03	16:19:50	Download: privesc.exe
20	16:37:44	16:37:46	SCP Command: scp -f /tmp/wpc*

Weak Recognition Measures:

Client Fingerprint:	Oracle Virtualbox with Several Open Ports
Browser Fingerprint:	Sufficient for PoC

Strong Recognition Measures:

Changed Linux Password:	False
Reuse of Unique Credentials:	True
Navigation to Unique URL:	True

Further Recognition Measures:

Usage of Downloaded Tools:	Weak Recognition (Publicly Available)
Building/Usage of Own Infrastructure:	Strong Recognition

Reliability of Recognition for each Session:

Confident:	4, 8, 9, 10, 12, 13, 17
Plausible:	2, 7, 11, 18, 19
Not Feasible:	1, 3, 5, 6, 14, 15, 16, 20

Attacker 3

First Login: 20:30:14 Last Logout: 21:58:00 Duration Attack: ~ 1.5 hours

Sessions: 8

1	20:30:14	20:30:15	Login (Tool)
2	20:30:28	20:31:20	Linux Commands: ls, id
3	20:31:42	20:31:45	Download: enum.sh
4	20:31:47	20:31:56	Linux Commands: ls
5	20:39:29	21:28:45	Attack Session: Enumeration, Privilege Escalation
6	21:28:51	21:32:29	Exploration: Website, Remmina
7	21:32:47	21:56:11	Attack Session: Lateral Movement
8	21:56:18	21:58:00	Validated Findings (CTF)

Weak Recognition Measures:

Client Fingerprint: Insufficient

Browser Fingerprint: Sufficient for PoC

Strong Recognition Measures:

Changed Linux Password: False

Reuse of Unique Credentials: True

Navigation to Unique URL: True

Further Recognition Measures:

Usage of Downloaded Tools: Weak Recognition (Publicly Available)

Reliability of Recognition for each Session:

Confident: 7, 8

Plausible: 3, 5

Not feasible: 1, 2, 4, 6

Attacker 4

First Login: 16:19:07 Last Logout: 16:44:45 Duration Attack: ~ 0.5 hours

Sessions: 8

1	16:19:07	16:20:20	Login & Password Change
2	16:20:22	16:23:12	Preparation for Honeypot-Check
3	16:23:30	16:23:35	Session Aborted
4	16:24:54	16:24:58	Honeypot-Check & Data Exfiltration
5	16:25:13	16:29:49	Exploration: Bash History, Remmina
6	16:29:56	16:32:27	Attempt to Navigate to injected URL
7	16:34:56	16:43:20	Exploration: Processes, Configuration Files
8	16:43:23	16:44:45	Validated Findings (CTF)

Weak Recognition Measures:

Client Fingerprint: Multiple Filterd Ports

Browser Fingerprint: Sufficient for PoC

Strong Recognition Measures:

Changed Linux Password: True

Reuse of Unique Credentials: False

Navigation to Unique URL: True

Further Recognition Measures:

Not Available

Reliability of Recognition for each Session:

Confident: 1, 2, 3, 4, 5, 6, 7, 8

Plausible: -

Not feasible: -

5.2 Evaluation of the Unknown Attackers

Except for the flags, the same architecture and topology are used for known and unknown attackers. In contrast to the known attackers, it is hard to predict the group of unknown attackers. Primarily, automated scanners and not very experienced attackers are expected. It is likely that they will focus on simple targets ("low-hanging fruits") and will not invest much time to compromise a system.
However, the approach of the thesis is designed for targeted attacks or in general for sophisticated attackers who are willing to spend time. The research setup allows multiple attackers to operate simultaneously with the limitation that only one user account is accessible on the Linux honeypot. Hence, the attackers will influence each other because, for example, tools and entries in the bash history can be accessed by anyone. Therefore, it was necessary to previously evaluate the attack volume targeting the honeypot to restrict the number of successful connections. In two short periods (one day and one week) all connection attempts including the tested credentials are logged. With this statistical information, credentials can be chosen to allow a specific number of attackers to connect to the honeypot.
Within one week a total of 60.708 connection attempts were recorded, that makes about 8.6725 connection attempts per day. In this period 766 unique usernames, 25.821 unique passwords, and 451 unique IP addresses were seen. Of the 451 IP addresses, 318 (about 70%) only made one single connection attempt. Interestingly, the username "admin" with the password "password" (from now on credentials are written as (<username>, <password>), e.g., (admin, password)) was most attempted by these 318 IP addresses and (root, root) was only tried three times. However, in total the most popular combination was (root, root) with 969 connection attempts.
In total, 10.610 attempts were recorded from one unique IP address, that makes about 17,5% of all connection attempts. Moreover, five "scanning clusters" (two to eight systems of a subnet) were detected. These clusters share password lists and probe in parallel. Table 5.3 presents the top 10 of tested usernames with the corresponding number of connection attempts and unique passwords. The username "root" is distinctly prominent and was used in about 91% of all connection attempts.

It can be assumed that there is a significant variance on every day; therefore, the second period of recordings were started for one single day. Within this day, a total of 16.411 connection attempts were logged. This is almost twice as much as the average number of connection attempts which were recorded in one week. On this day, 47 unique IP addresses, 93 unique usernames, and 9.479 unique passwords were seen. One of these IP addresses made 3.067 connection attempts, which is about 18,5% of all attempts. Furthermore, three clusters (four to eight systems of a subnet) were detected, and interestingly all these

Username	Conn. Attempts	Unique Passwords
root	55.324	24.936
admin	540	128
nagios	181	74
test	179	99
guest	140	54
oracle	90	39
zabbix	77	23
user	65	24
ubnt	62	16
git	58	31

Table 5.3: Top 10 usernames of SSH connection attempts

systems were seen during the testing period of one week. This indicates that the clusters share password lists and workload, though they do not store processed IP addresses. This insight could be relevant in the case of reflection if it is necessary to change the IP address of a honeypot regularly. At least for automated attacks against SSH, it is not necessary. Table 5.4 below shows the three noticeable clusters.

Cluster "Shanghai"	Cluster "Zhejiang"	Cluster "Jiangsu"
221.194.44.211	115.238.245.2	122.226.181.164
221.194.44.232	115.238.245.4	122.226.181.165
221.194.47.205	115.238.245.6	122.226.181.166
221.194.47.221	115.238.245.8	122.226.181.167
221.194.47.233	115.238.245.14	
221.194.47.236		
221.194.47.239		
221.194.47.243		

Table 5.4: Detected scanning clusters

The clusters are named by the region from which the IP addresses allegedly originates. It is important to emphasize that this is not an attribution and this does not indicate that the attackers are residing there. An IP geolocation is not reliable; for instance, it might be a VPN service, a proxy system, a rented server, or a bot system. In this case, the region is just used as a name for the corresponding cluster.

Since a weak password is chosen as the vulnerability of the Linux honeypot, it will be orientated to the information of the table 5.3. It is necessary for the analysis to minimize the mutual influence of multiple random attackers. Also, the usernames “root” and “admin” do not match the properties of the honeypot. The attacker will access an unprivileged user account of a common Ubuntu Linux system. Therefore, “root” and “admin” are excluded as they raise certain expectations of user privileges and would immediately arouse mistrust if their privileges are limited. The user “nagios” raises similar expectations. This user account suggested a monitoring system with appropriate privileges. However, the honeypot does not offer the infrastructure for a monitoring system and, as mentioned before, only provides an unprivileged user account. Consequently, the username “test” is chosen as it fits well in the given scenario: A test/maintenance system accessible from the Internet with a misconfiguration allowing a transition into the internal network.
Table 5.5 shows the number of connection attempts of the top 10 passwords for the username “test”. Since the evaluation period for unknown attackers was planned for one week, the password “test123” is selected. This credentials statistically correspond to approximately one successfully established connection a day. With regard to the uncertainty of random attackers, this seems to be appropriate as the cross-influencing of multiple attackers can lead to destructive behavior in order to sabotage the alleged rival or the victim system.

Password	**Seen**
test	19
test123	8
1234	6
password	6
12345	6
123456	6
test1234	5
testtest	5
111111	4
123	4

Table 5.5: Top 10 passwords for username “test”

Results

A challenging evaluation of the approach of the thesis is expected for random attackers. For a successful recognition, the attacker must have certain skills and must spend time on the system. The concept is designed for targeted attacks and sophisticated attack-

ers. The likelihood is small that such a candidate will be captured during the evaluation period. Primarily automatic scans and attackers aiming for simple targets are expected. Presumably, such attackers want to plant malware on the victim's system, for instance, to integrate it into a botnet or misuse the computing resources for mining cryptocurrencies. None of this is easily achievable on the provided honeypot. Therefore, the honeypot is not a simple target which is also not desired concerning the objectives of this thesis.

During the evaluation period, the honeypot was accessible via the Internet. The SSH server can be reached on port 22. For a successful authentication the username "test" and the password "test123" were chosen. These credentials were not arbitrarily but statically selected on the basis of the previous research.
At first, the honeypot was accessible for one week. Within this period 56.881 connection attempts were recorded; 7 attempts from 6 unique IP addresses have been successful. The IP address found twice processed the same password list of 105 entries and has behaved entirely identically. A quick check of the IP addresses revealed that all of them are already blacklisted [84]: One in December 2017, one in June 2018 and four recently in July 2018. Table 5.6 presents the successful connection attempts with the relevant parameters (date and time, IP address and attempts to log in).

No.	Date and Time	IP Address	Connection Attempts
1	July 19, 2018 10:38:30	199.19.226.63	85
2	July 21, 2018 01:40:28	41.33.125.30	1.253
3	July 22, 2018 05:58:38	132.148.141.155	7
4	July 22, 2018 09:37:54	190.121.17.82	210
5	July 22, 2018 09:49:35	62.210.220.115	105
6	July 22, 2018 10:47:51	80.211.62.234	6
7	July 23, 2018 06:42:40	62.210.220.115	105

Table 5.6: Overview of the successful logins of random attackers

It is noticeable that all attackers except *Attacker 2* processed short and specific password lists. Mainly default and standard credentials of several services were tested but "root" was negligible. In total, 5 of the 7 sessions were without any interaction. After the automatic brute force tool successfully established a connection, it disconnected immediately. Most likely, the remaining two attackers also used an automated attack (script) but customized it to gather additional information from the system. *Attacker 2* displayed the user and group information by using the *id* [63] command. Then, after about 1.5 seconds he disconnected. This short period of time and the many failed connection attempts right before indicate automation without any human interaction. *Attacker 3* executed 5 commands in

approximately 5 seconds. This similar behavior (failed connection attempts, short period of time) also suggests an automatic script even though it is a little smarter. At first, he ensured that there was no history of his activities by executing the following:

> "unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH ; history -n ; export HISTFILE=/dev/null ; export HISTSIZE=0; export HISTFILESIZE=0;"

Then he read several system information such as all processes of the current user (*ps -x* [110]), details about logical processors of the system (*/proc/cpuinfo*) and free/used memory capacities (*free -m* [48]).
It seems that both attackers used this process as an automated information gathering to find potentially profitable targets. Since *Attacker 3's* visit looked very promising, the evaluation period was extended by another week.
The second evaluation week provided nearly the same results as the first week: 34.720 connection attempts, 6 successful logins from 5 unique IP addresses. All sessions except from one session were without any interaction. The only command executed on the honeypot was *uname -a* [150] to print system informations. None of the IP addresses were seen before but one IP address (80.211.62.32) seems to be from a subnet seen the week before. However, no attacker who gathered information in the first evaluation week was seen again. The reasons could be, for example, that the honeypot was not a profitable target or they have not had the time.
However, none of the collected information of the 11 attackers is sufficient to recognize them. This is explained by the factors of a maximal connection time of 5 seconds and no human interaction.

The results clearly show how difficult it is to evaluate the approach of the thesis with random real-world attackers. This is due to the concept which is designed for targeted attacks and less for automated attacks performed by haphazard scanners. Other factors could be the IP address of the honeypot since it is located in the German Research Network and the limited capabilities and computing resources of the VMs. This could deter the attackers or declare the systems as unattractive. Especially with regard to the currently aggressively spread cryptomining malware. Nevertheless, this kind of attacker is irrelevant for the approach since they aim for simple targets and do not necessarily compromise a system or perform lateral movement.
A decent position for a reliable evaluation would be in a DMZ of a larger company that is targeted attacked by sophisticated attackers. However, on the one hand, a suitable research environment is missing, and on the other hand, this would be unpredictable in terms of time. It can therefore only be concluded that no statement can be made regarding the effectiveness of the approach for random attackers. This is based on the

different aims being pursued by these kinds of attackers (low-hanging fruits, automatic scripts). Otherwise, an unspecified length for the evaluation period is needed until a decent attack has taken place.

Chapter 6

Conclusion and Future Work

In the course of the thesis, an approach based on honeypots was developed to recognize sophisticated attackers in a network. The implemented solution is a complex architecture including a transparent proxy, an (entry point/stepping stone) honeypot, and the HoneyLab. This architecture as well as several measures for recognition were developed, a PoC was implemented and finally evaluated.
The following conclusion is drawn based on known attackers (security professionals) since no sophisticated attacker was captured in the evaluation period. On the one hand, the control group of known attackers is small and differs from the testing group of random attackers (skill set and invested time), on the other hand, their TTP can be applied to sophisticated attackers.

6.1 Conclusion

The concept behind the approach of the thesis is to let the attacker build up knowledge and inject unique information in his knowledge. By performing his cyber kill chain (for example Reconnaissance, Intrusion, Exploitation, Privilege Escalation, Lateral Movement, and Exfiltration), the attacker goes through a learning process which is influenced in such a way that his further activities leave a unique fingerprint.
The results of the evaluation showed that every known attacker could subtly be manipulated. Injected credentials are an effective measure to trace malicious activities in the network and link particular sessions. Only changing the weak initial password to a complex password was even better since it made any session of this attacker traceable. Unfortunately, it was not changed as often as another research suggested [2]. Even the injected URLs worked well and yielded sufficient results for the PoC.
It may be argued that security professionals such as penetration testers may behave differently than sophisticated attackers in an APT since the first group rarely attach importance to covert operations. However, the approach of the thesis focuses on the process of infor-

mation gathering to compromise the gained knowledge. Considering the cyber kill chain mentioned above, the approach affects the first two phases: Reconnaissance and Intrusion. These phases are identical for both groups (security professionals and sophisticated attackers) and do not have significant differences. The Reconnaissance contains several aspects of information gathering and enumeration of, for example, services. The aim is to collect as much information as possible about the target (passively and actively). Except for the active measure of the approach (enhanced port and service scan of the attacker's client), all measures are designed to allow an attacker to find them in the first stage of his attack. This directly influences the next phase (Intrusion) where the attacker can unconsciously be guided through the information. For instance, targets can now be predefined by the defender to evoke, to some extent, a deterministic behavior. The remaining phases of the cyber kill chain may differ for various groups but are not relevant to the approach of this thesis. Due to the Intrusion based on the Reconnaissance, it should be able to recognize the attacker already.
The assumption that any sophisticated attackers could be recognized is confirmed by the evaluation. Furthermore, it was noticed that the more skilled an attacker is, the more knowledge could be compromised. This is due to the fact that he spent more time on the systems and gathered more information.

Unfortunately, a reliable attribution is not feasible by using the gathered information. The approach of the thesis yields strong recognition features, but these cannot be used for attribution. Only the controlled website can reveal parts of the attacker's infrastructure or his system/location when he makes a gross mistake by forgetting a VPN, proxy or stepping stone system.

Finally, it can be noted that the approach of the thesis can confirm the hypothesis: It is possible to detect and, furthermore, to recognize a specific adversary by using appropriate measures based on honeypots.

6.2 Future Work

The PoC offers several possibilities for expansion where existing modules can be optimized or completely new components can be developed. Furthermore, the evaluation of the implemented approach can be repeated, for example, in a DMZ of a company to put the setup in the right context. This could improve the findings and informative value for real-world attackers.

For an application in practice, the entire process can be automated. Currently, the analysis has to be done manually, but it should be simple to automate. For this purpose, the

"Elastic Stack" can be used which components are already partly considered or implemented. For instance, the output of Sysdig or Falco can be processed directly and also the self-developed logging format should be simple to integrate and parse due to the consistent and straightforward design.
Furthermore, the (Linux) honeypot can be enhanced. There are not many free/open-source high-interaction honeypots publicly available, and the PoC still allows optimizations in several aspects. For example, the monitoring and hiding functionality can be improved, or further techniques such as anti-VM or anti-sandbox detection can be implemented.

Moreover, the approach of the thesis can be enhanced. At first, browser fingerprinting can be improved since it is rudimentarily used in the PoC. For recognition navigation to the injected URL is sufficient; however, this discards great opportunities for further information retrieval. Then, the website can be replaced by a web version of the HoneyLab. For instance, an entire bogus company website or even a vulnerable CMS will provide more possibilities for information acquisition from unknown threats, zero-day exploits, and adversarial TTP.
Additionally, other passive or active measures can be implemented such as executing code in the attacker's browser or a real-time manipulation of files which are downloaded by the attacker. For example, tracking pixels or arbitrary macros could be injected in particular files. Of course, this needs to be reviewed from a legal perspective but could be of interest for research.

Finally, new recognition features can be developed which also allow an attribution. These must be more general and cannot exclusively/directly refer to the session or architecture. The initial password could be such a feature when the attacker makes a mistake and reuses a complex password on multiple independent attacks. However, there are many more features to evaluate.

Appendix A

Username	Password	Encrypted Password
smanbrow	lakers	mncZwbzo1As=
smankalt	summer2018	WFj8gIl8vDT3zSbkhlOibw==
smchmend	welcome	17Vrh0xbe0s=
smchspen	1q2w3e	Gb0YjKIG7qc=
smdewill	harley	dpbIVOQd13E=
smigfrec	cookie	gen1H6F/kss=
smjohern	password1	I8TaoGrFMLnbxDvleB8Bug==
smjosmit	soccer	SYDDXZor8Qg=
smjuherr	mercedes	bYLux3ENJ42SItM59vpJHg==
smjureit	football	d780jfgKaWwdALQ9IMkplg==
smkahayn	iloveyou	XdIjTCCeu5UCJtBT8XTeaw==
smkeborg	passw0rd	i16DUtgvp/EUMXABzWi43Q==
smlesant	jessica	d90Zfm4nxf0=
smmajurg	123456	zZgZCDE1Y5k=
smmazell	london	CK3JGWHduj0=
smmewatt	password	I8TaoGrFMLmH4n5QODhfMQ==
smmidani	cheese	o8sPs+qWFQU=
smmipars	banana	CZgI+DqolgQ=
smremull	freedom	QL50Ae/4EpM=
smsibles	letmein	iLWlCOn2iaU=
smtajone	whatever	jwmEOoROwFJffmfW8526aQ==
smtifuch	qwerty	Gd7M3pZe7V4=
smtinagl	1qaz2wsx	scwNIqD5vaHB4lQjVNYlUQ==
smulheck	654321	8CgAeK6SoBo=
smwatroc	trustno1	NMhhg1eEm89k1ORsiwKaDA==

Table A.1: Credentials with 3DES encrypted passwords for remmina

URL Paths
?q=testadmin
_media
admin/login
administrator
app/kibana/dashboard/marketing
app/kibana/dashboard/presales
configuration.php
data/exchange
dropbox/configuration.php
intranet/config.php
intranet/login.html
intranet/wp-login.php
mail/ecp
mailman/listinfo/mbinfo.html
owa
owa/remote
phpmyadmin
upload/index.php
user/login
webdev/index.html
wip/apache/config
wip/mysql/config
wp-admin/
wp-login.php
wp/wp-login.php

Table A.2: Conceivably tempting paths for arbitrary domains

List of Figures

Listings

Bibliography

[1] ADDUSER(8) - System Manager's Manual. *adduser, addgroup - Add a User or Group to the System.* URL `https://linux.die.net/man/8/adduser`. Last accessed: August 02, 2018. [man adduser].

[2] Alata, Eric and Nicomette, Vincent and Kaâniche, Mohamed and Dacier, Marc and Herrb, Matthieu. Lessons Learned from the Deployment of a High-Interaction Honeypot. In *Dependable Computing Conference, 2006. EDCC'06. Sixth European*, pages 39–46. IEEE, 2006.

[3] Alexander, Harriet and Ryall, Julian. Russian Hackers Posed as North Koreans to Launch Cyberattack on Winter Olympics, Claims US, February 2018. URL `https://www.telegraph.co.uk/news/2018/02/25/russian-hackers-pose-north-koreans-launch-cyberattack-winter/`. Last accessed: August 02, 2018.

[4] Alrabaee, Saed and Saleem, Noman and Preda, Stere and Wang, Lingyu and Debbabi, Mourad. Oba2: An Onion Approach to Binary Code Authorship Attribution. *Digital Investigation*, 11:S94–S103, 2014.

[5] Banks, Andrew and Gupta, Rahul. MQTT Version 3.1.1. *OASIS Standard*, October 2014. URL `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf`. Last accessed: August 02, 2018.

[6] Barth, Adam. HTTP State Management Mechanism. April 2011. URL `https://tools.ietf.org/html/rfc6265`. Last accessed: August 02, 2018.

[7] Bartholomew, Brian and Guerrero-Saade, Juan Andres. Wave Your False Flags! Deception Tactics Muddying Attribution in Targeted Attacks, 2016. URL `https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2017/10/20114955/Bartholomew-GuerreroSaade-VB2016.pdf`. Last accessed: August 02, 2018.

[8] Beuth, Patrick and Gebauer, Matthias. Hackergruppe Snake: Vor Diesen Elite-Hackern Ist Kein Land der Welt Sicher, March 2018. URL `https://www.watson.ch/!894325670`. Last accessed: August 02, 2018.

[9] BlackHatAcademy.org. JynxKit2, October 2011. URL `https://github.com/chokepoint/Jynx2`. Last accessed: August 02, 2018.

[10] Botezatu, Bogdan. New Hide 'N Seek IoT Botnet Using Custom-built Peer-to-Peer Communication Spotted in the Wild, January 2018. URL `https://labs.bitdefender.com/2018/01/new-hide-n-seek-iot-botnet-using-custom-built-peer-to-peer-communication-spotted-in-the-wild`. Last accessed: August 02, 2018.

[11] Browserleaks. Web Browser Security Checklist for Identity Theft Protection, 2011. URL `https://browserleaks.com/`. Last accessed: August 02, 2018.

[12] Canonical Ltd. Ubuntu - Alternative Downloads, March 2018. URL `https://www.ubuntu.com/download/alternative-downloads`. Last accessed: August 02, 2018.

[13] Cao, Yinzhi and Li, Song and Wijmans, Erik. Browser Fingerprinting Via OS and Hardware Level Features. In *Proceedings of Network & Distributed System Security Symposium (NDSS)*, 2017.

[14] Chairetakis, Eleftherios and Alkudhir, Bassam and Mystridis, Panagiotis. Deployment of Low Interaction Honeypots in University Campus Network, 2013.

[15] Cheswick, Bill. An Evening with Berferd in which a Cracker is Lured, Endured, and Studied. In *Proc. Winter USENIX Conference, San Francisco*, pages 20–24, 1992.

[16] Childs, Dustin. The Results - Pwn2Own 2017 Day Three, March 2017. URL `https://www.zerodayinitiative.com/blog/2017/3/17/the-results-pwn2own-2017-day-three`. Last accessed: August 02, 2018.

[17] Cohen, Fred and others. The Deception Toolkit. *Risks Digest*, 19, 1998. URL `http://www.all.net/dtk`. Last accessed: March 18, 2018.

[18] Cole, Eric and Northcutt, Stephen. Honeypots: A Security Manager's Guide to Honeypots. URL `https://www.sans.edu/cyber-research/security-laboratory/article/honeypots-guide`. Last accessed: August 02, 2018.

[19] Corey, Joseph. Advanced Honey Pot Identification, 2004. URL `http://www.ouah.org/p63-0x09.txt`. Last accessed: August 02, 2018.

[20] Crowdstrike Global Intelligence Team. CrowdStrike Intelligence Report: Putter Panda, 2014. URL `https://cdn0.vox-cdn.com/assets/4589853/crowdstrike-`

`intelligence-report-putter-panda.original.pdf`. Last accessed: August 02, 2018.

[21] Dargin, Mark. Increase Your Network Security: Deploy a Honeypot, October 2017. URL `https://www.networkworld.com/article/3234692/lan-wan/increase-your-network-security-deploy-a-honeypot.html`. Last accessed: August 02, 2018.

[22] de Maizière, Thomas. "Deutschland hat mit dem IVBB eines der sichersten Regierungsnetzwerke der Welt!", March 2018. URL `https://www.bmi.bund.de/SharedDocs/kurzmeldungen/DE/2018/03/statement-cyberangriff-ivbb.html`. Last accessed: August 02, 2018.

[23] Degioanni, Loris. Sysdig Falco - Behavioral Activity Monitoring With Container Support, April 2018. URL `https://github.com/draios/falco`. Last accessed: August 02, 2018.

[24] Degioanni, Loris, et al. The Sysdig Container Toolkit - Open Source Troubleshooting, Forensics, and Security, March 2018. URL `https://github.com/draios/sysdig`. Last accessed: August 02, 2018.

[25] DLOPEN(3) - Linux Programmer's Manual. *dlclose, dlopen, dlmopen - Open and Close a Shared Object*, September 2015. URL `https://linux.die.net/man/3/dlopen`. Last accessed: August 02, 2018. [man dlopen].

[26] DLSYM(3) - Linux Programmer's Manual. *dlsym, dlvsym - Obtain Address of a Symbol in a Shared Object or Executable*, September 2015. URL `https://linux.die.net/man/3/dlsym`. Last accessed: August 02, 2018. [man dlsym].

[27] Dornseif, Maximillian and Holz, Thorsten and Klein, Christian N. NoSEBrEaK - Attacking Honeynets. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 123–129. IEEE, 2004.

[28] Eckersley, Peter. How Unique Is Your Web Browser?, 2010. URL `https://panopticlick.eff.org/`. Last accessed: August 02, 2018.

[29] Edwards, C.I.P.M. An Analysis of a Cyberattack On a Nuclear Plant: The Stuxnet Worm. *Critical Infrastructure Protection*, 116:59, 2014.

[30] Ehrsam, William F. and Meyer, Carl H. W. and Smith, John L. and Tuchman, Walter L. Message Verification and Transmission Error Detection by Block Chaining, February 1978. US Patent 4,074,066.

[31] Elasticsearch BV. Beats - Lightweight Data Shippers, February 2018. URL `https://www.elastic.co/products/beats`. Last accessed: August 02, 2018.

[32] Elasticsearch BV. Elasticsearch - Search, Analyze, and Store Your Data, April 2018. URL `https://www.elastic.co/products/elasticsearch`. Last accessed: August 02, 2018.

[33] Elasticsearch BV. Filebeat - Lightweight Shipper for Logs, April 2018. URL `https://www.elastic.co/products/beats/filebeat`. Last accessed: August 02, 2018.

[34] Elasticsearch BV. Kibana - Visualize Your Data. Navigate the Elastic Stack, April 2018. URL `https://www.elastic.co/products/kibana`. Last accessed: August 02, 2018.

[35] Elasticsearch BV. Logstash - Centralize, Transform and Stash Your Data, April 2018. URL `https://www.elastic.co/products/logstash`. Last accessed: August 02, 2018.

[36] Elasticsearch BV. Elastic - Powering Data Search, Log Analysis, Analytics, April 2018. URL `https://www.elastic.co/products`. Last accessed: August 02, 2018.

[37] F-Secure. Incident Response Report, 2018. URL `https://fsecurepressglobal.files.wordpress.com/2018/02/f-secure-incident-response-report.pdf`. Last accessed: August 02, 2018.

[38] D. P. Fidler. Was stuxnet an act of war? decoding a cyberattack. *Security & Privacy, IEEE*, 9(4).

[39] R. Fielding, M. Nottingham, and J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Caching. June 2014. URL `https://tools.ietf.org/html/rfc7234`. Last accessed: August 02, 2018.

[40] Fielding, Roy and Reschke, Julian. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. June 2014. URL `https://tools.ietf.org/html/rfc7230`. Last accessed: August 02, 2018.

[41] Fielding, Roy and Reschke, Julian. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. June 2014. URL `https://tools.ietf.org/html/rfc7231`. Last accessed: August 02, 2018.

[42] Fielding, Roy and Reschke, Julian. Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. June 2014. URL `https://tools.ietf.org/html/rfc7232`. Last accessed: August 02, 2018.

[43] Fielding, Roy and Reschke, Julian. Hypertext Transfer Protocol (HTTP/1.1): Authentication. June 2014. URL `https://tools.ietf.org/html/rfc7235`. Last accessed: August 02, 2018.

[44] FIND(1) - General Commands Manual. *find - Search for Files in a Directory Hierarchy*, December 2015. URL `https://linux.die.net/man/1/find`. Last accessed: August 02, 2018. [man find].

[45] FireEye, Inc. M-Trends 2017: A View from the Front Lines, March 2017. URL `https://www2.fireeye.com/rs/848-DID-242/images/RPT-M-Trends-2017.pdf`. Last accessed: August 02, 2018.

[46] Foreign & Commonwealth Office, National Cyber Security Centre, and Lord Ahmad of Wimbledon. Foreign Office Minister Condemns Russia for NotPetya Attacks, February 2018. URL `https://www.gov.uk/government/news/foreign-office-minister-condemns-russia-for-notpetya-attacks`. Last accessed: August 02, 2018.

[47] Frantzeskou, Georgia and Gritzalis, Stefanos and MacDonell, Stephen G. Source Code Authorship Analysis for Supporting the Cybercrime Investigation Process. *Handbook of Research on Computational Forensics, Digital Crime, and Investigation: Methods and Solutions*, pages 470–495, 2004.

[48] FREE(1) - User Commands. *free - Display Amount of Free and Used Memory in the System*, July 2014. URL `https://linux.die.net/man/1/free`. Last accessed: August 02, 2018. [man free].

[49] Galeotti, Mark. *Global Crime Today: The Changing Face of Organised Crime*. Routledge, 2014.

[50] Gatta, Antenore and Panozzo, Giovanni and Cavedon, Dario. Remmina - A Feature Rich Remote Desktop Application, October 2009. URL `https://remmina.org/`. Last accessed: August 02, 2018.

[51] Gierow, Hauke. JenX - IoT-Botnetz mit Zentralen Steuerservern Entdeckt, February 2018. URL `https://www.golem.de/news/jenx-iot-botnetz-mit-zentralen-steuerservern-entdeckt-1802-132567.html`. Last accessed: August 02, 2018.

[52] Gierow, Hauke and Greis, Friedhelm. Bundeshack: Ausländischer Geheimdienst Soll Regierung Gewarnt Haben, March 2018. URL `https://www.golem.de/news/bundeshack-auslaendischer-geheimdienst-soll-regierung-gewarnt-haben-1803-133112.html`. Last accessed: August 02, 2018.

[53] Göbel, Jan Gerrit and Dewald, Andreas. *Client-Honeypots: Exploring Malicious Websites*. Oldenbourg Verlag, 2011.

[54] Goncharov, Max. Russian Underground 101. *Trend Micro Incorporated Research Paper*, page 51, 2012.

[55] Guerrero-Saade, Juan Andres and Raiu, Costin. Walking in Your Enemy's Shadow: When Fourth-Party Collection Becomes Attribution Hell, 2017. URL `https://media.kasperskycontenthub.com/wp-content/uploads/sites/31/2017/10/22065248/Guerrero-Saade-Raiu-VB2017.pdf`. Last accessed: August 02, 2018.

[56] Hacker Target. Cowrie Honeypot Analysis (24 hours of Attacks), March 2018. URL `https://hackertarget.com/cowrie-honeypot-analysis-24hrs/`. Last accessed: August 02, 2018.

[57] Halder, Debarati and Jaishankar, Karuppannan. *Cyber Crime and the Victimization of Women: Laws, Rights and Regulations*. IGI Global, 2011.

[58] Holz, Thorsten and Raynal, Frederic. Detecting Honeypots and Other Suspicious Environments. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 29–36. IEEE, 2005.

[59] Horejsi, Jaromir. Campaign Possibly Connected to "MuddyWater" Surfaces in the Middle East and Central Asia, March 2018. URL `https://blog.trendmicro.com/trendlabs-security-intelligence/campaign-possibly-connected-muddywater-surfaces-middle-east-central-asia`. Last accessed: August 02, 2018.

[60] Hubain, Charles. LD_NOT_PRELOADED_FOR_REAL, February 2015. URL `http://haxelion.eu/article/LD_NOT_PRELOADED_FOR_REAL/`. Last accessed: August 02, 2018.

[61] Hunt, Troy. Have I Been Pwned: Pwned Passwords, March 2018. URL `https://haveibeenpwned.com/Passwords`. Last accessed: August 02, 2018.

[62] IBM Security. IBM X-Force Threat Intelligence Index 2017, March 2017.

[63] ID(1) - User Commands. *id - Print Real and Effective User and Group IDs*, February 2017. URL `https://linux.die.net/man/1/id`. Last accessed: August 02, 2018. [man id].

[64] Ierusalimschy, Roberto and Celes, Waldemar and De Figueiredo, Luiz Henrique. Lua - The Programming Language, 1993. URL `https://www.lua.org/`. Last accessed: August 02, 2018.

[65] Information Security Stack Exchange. Is It Possible to Detect a Honeypot? [closed], 2015. URL `https://security.stackexchange.com/questions/90642/is-it-possible-to-detect-a-honeypot`. Last accessed: August 02, 2018.

[66] Innes, Simon and Valli, Craig. Honeypots: How Do You Know When You Are Inside One? In *Australian Digital Forensics Conference*, page 28, 2006.

[67] Javelin Networks. Honeypot Buster: Detect Honeypots and Lures. Empower Red Teams., 2017. URL `https://github.com/JavelinNetworks/HoneypotBuster`. Last accessed: August 02, 2018.

[68] Josefsson, Simon. The Base16, Base32, and Base64 Data Encodings. October 2006. URL `https://tools.ietf.org/html/rfc4648`. Last accessed: August 02, 2018.

[69] Kamkar, Samy. Evercookie., September 2010. URL `https://samy.pl/evercookie/`. Last accessed: August 02, 2018.

[70] Karn, Phil and Metzger, Perry and Simpson, William Allen. The ESP Triple DES Transform. September 1995. URL `https://tools.ietf.org/html/rfc1851`. Last accessed: August 02, 2018.

[71] KILLALL(1) - User Commands. *killall - Kill Processes by Name*, June 2017. URL `https://linux.die.net/man/1/killall`. Last accessed: August 02, 2018. [man killall].

[72] Kocher, Paul and Genkin, Daniel and Gruss, Daniel and Haas, Werner and Hamburg, Mike and Lipp, Moritz and Mangard, Stefan and Prescher, Thomas and Schwarz, Michael and Yarom, Yuval. Spectre Attacks: Exploiting Speculative Execution. *ArXiv e-prints*, January 2018.

[73] Kunkel, Matthias. [ilAdmins] ILIAS und die Cyber-Attacke auf das Datennetz der Bundesregierung, March 2018. URL `https://lists.ilias.de/pipermail/ilias-admins/2018-March/000066.html`. Last accessed: August 02, 2018.

[74] Lafon, Yves and Fielding, Roy and Reschke, Julian. Hypertext Transfer Protocol (HTTP/1.1): Range Requests. June 2014. URL `https://tools.ietf.org/html/rfc7233`. Last accessed: August 02, 2018.

[75] Layton, Robert and Watters, Paul A. *Automating Open Source Intelligence: Algorithms for OSINT*. Syngress, 2015.

[76] LDD(1) - Linux Programmer's Manual. *ldd - Print Shared Object Dependencies*, September 2017. URL `https://linux.die.net/man/1/ldd`. Last accessed: August 02, 2018. [man ldd].

[77] LD.SO(8) - Linux Programmer's Manual. *ld.so, ld-linux.so - Dynamic Linker/Loader*, September 2017. URL `https://linux.die.net/man/8/ld.so`. Last accessed: August 02, 2018. [man ld.so].

[78] LIBC(7) - Linux Programmer's Manual. *libc - Overview of Standard C Libraries on Linux*, December 2016. URL `https://linux.die.net/man/7/libc`. Last accessed: August 02, 2018. [man libc].

[79] linuxnewb and Reifschneider, Sean and Foerster, David and other. How to Extract Saved Password From Remmina?, May 2013. URL `https://askubuntu.com/questions/290824/how-to-extract-saved-password-from-remmina`. Last accessed: August 02, 2018.

[80] Lipp, Moritz and Schwarz, Michael and Gruss, Daniel and Prescher, Thomas and Haas, Werner and Mangard, Stefan and Kocher, Paul and Genkin, Daniel and Yarom, Yuval and Hamburg, Mike. Meltdown. *ArXiv e-prints*, January 2018.

[81] LOGIN(1) - User Commands. *login - Begin Session on the System*, June 2012. URL `https://linux.die.net/man/1/login`. Last accessed: August 02, 2018. [man login].

[82] LS(1) - User Commands. *ls - List Directory Contents*, December 2017. URL `https://linux.die.net/man/1/ls`. Last accessed: August 02, 2018. [man ls].

[83] Lyon, Gordon and other. Nmap: the Network Mapper, September 1997. URL `https://nmap.org/`. Last accessed: August 02, 2018.

[84] Marathon Studios, Inc. AbuseIPDB - Making the Internet Safer, One IP at a Time), March 2016. URL `https://www.abuseipdb.com/`. Last accessed: August 02, 2018.

[85] Matherly, John. SHODAN - The Computer Search Engine, 2009. URL `https://www.shodan.io/`. Last accessed: August 02, 2018.

[86] McAfee, LLC. 2017 Threats Predictions, November 2016. URL `https://www.mcafee.com/us/resources/reports/rp-threats-predictions-2017.pdf`. Last accessed: August 02, 2018.

[87] McGrew, Robert. Experiences with Honeypot Systems: Development, Deployment, and Analysis. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 9, pages 220a–220a. IEEE, 2006.

[88] McWhorter, Dan. Mandiant Exposes APT1 - One of China's Cyber Espionage Units & Releases 3,000 Indicators. *Mandiant, February*, 18, 2013.

[89] Meer, Haroon and Slaviero, Marco. Bring Back the Honeypots, 2015. URL `https://www.blackhat.com/us-15/briefings.html#bring-back-the-honeypots`. Last accessed: August 02, 2018.

[90] Microsoft; Designed by Snover, Jeffrey and Payette, Bruce and Truher, James (et al.). PowerShell Documentation), November 2006. URL `https://docs.microsoft.com/en-us/powershell/`. Last accessed: August 02, 2018.

[91] Miessler, Daniel. Information Security Concepts, June 2014. URL `https://danielmiessler.com/study/infosecconcepts/`. Last accessed: August 02, 2018.

[92] Miessler, Daniel and Haddix, Jason. SecLists - The Pentester's Companion, March 2018. URL `https://github.com/danielmiessler/SecLists/tree/master/Passwords`. Latest Commit: July 12, 2018. Commit-ID: 9c67af9639d6ee849bc02fec163d6f336097a174. Last accessed: August 02, 2018.

[93] Mokube, Iyatiti and Adams, Michele. Honeypots: Concepts, Approaches, and Challenges. In *Proceedings of the 45th Annual Southeast Regional Conference*, pages 321–326. ACM, 2007.

[94] Morris, Andrew. Quick Proof of Concept to Detect a Kippo SSH Honeypot Instance Externally, 2014. URL `https://github.com/andrew-morris/kippo_detect`. Last accessed: August 02, 2018.

[95] National Research Council and others. *Technology, Policy, Law, and Ethics Regarding US Acquisition and Use of Cyberattack Capabilities*. National Academies Press, 2009.

[96] NDR/ARD-Aktuell, Tagesschau. Cyberattacke Bestätigt - Bundesregierung Wurde Gehackt, February 2018. URL `http://www.tagesschau.de/inland/hackerangriff-regierungsnetz-101.html`. Last accessed: August 02, 2018.

[97] Netscape Communications Corporation, Mozilla Foundation, Ecma International; Designed by Eich, Brendan. JavaScript (Programming Language, 1995. URL `https://developer.mozilla.org/en-US/docs/Web/JavaScript`. Last accessed: August 02, 2018.

[98] Nicholson, Thomas. HonSSH, December 2016. URL `https://github.com/tnich/honssh`. Latest Release: 1.1.2 on April 4, 2018. Commit-ID: 7adbf1bad824e852f4fe03ea0d01eaae2e6758e3. Last accessed: August 02, 2018.

[99] Nirkhi, Smita and Dharaskar, Rajiv V. Comparative Study of Authorship Identification Techniques for Cyber Forensics Analysis. *arXiv preprint arXiv:1401.6118*, 2013.

[100] OPC Foundation. OPC Unified Architecture Specification, 2018. URL `https://opcfoundation.org/developer-tools/specifications-unified-architecture`. Last accessed: August 02, 2018.

[101] Oracle. Oracle VM VirtualBox, May 2018. URL `https://www.virtualbox.org/`. Last accessed: August 02, 2018.

[102] Oracle Corporation. Chapter 6. Virtual networking. URL `https://www.virtualbox.org/manual/ch06.html`. Last accessed: August 02, 2018.

[103] PAM(8) - Linux-PAM Manual. *PAM, pam - Pluggable Authentication Modules for Linux*, April 2016. URL `https://linux.die.net/man/8/pam`. Last accessed: August 02, 2018. [man PAM].

[104] PAM_AUTHENTICATE(3) - Linux-PAM Manual. *pam_authenticate - Account Authentication*, April 2016. URL `https://linux.die.net/man/3/pam_authenticate`. Last accessed: August 02, 2018. [man pam_authenticate].

[105] PAM_CHAUTHTOK(3) - Linux-PAM Manual. *pam_chauthtok - Updating Authentication Tokens*, April 2016. URL `https://linux.die.net/man/3/pam_chauthtok`. Last accessed: August 02, 2018. [man pam_chauthtok].

[106] Parker, Tom. Stuxnet Redux: Malware Attribution & Lessons Learned. *Black Hat DC*, 2011. URL `www.blackhat.com/html/bh-dc-11/bh-dc-11-archives.html#Parker`. Last accessed: August 02, 2018.

[107] PASSWD(1) - User Commands. *passwd - Update User's Authentication*, December 2017. URL `https://linux.die.net/man/1/passwd`. Last accessed: August 02, 2018. [man passwd].

[108] Pierluigi Paganini. Zeus, Software as a Service – Implications for Civil and Military, January 2012. URL `http://securityaffairs.co/wordpress/1971/cybercrime/zeussoftware-as-a-service-implications-for-civil-and-military.html`. Last accessed: August 02, 2018.

[109] Pouget, Fabien and Dacier, Marc and Debar, Hervé. White Paper: Honeypot, Honeynet, Honeytoken: Terminological Issues. *Rapport technique EURECOM*, 1275, 2003.

[110] PS(1) - User Commands. *ps - Report a Snapshot of the Current Processes*, August 2015. URL `https://linux.die.net/man/1/ps`. Last accessed: August 02, 2018. [man ps].

[111] Python Software Foundation; Designed by Van Rossum, Guido. Python (Programming Language), 1991. URL `https://www.python.org/`. Last accessed: August 02, 2018.

[112] Qassrawi, Mahmoud T. and Zhang, Hongli. Client Honeypots: Approaches and Challenges. In *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, pages 19–25. IEEE, 2010.

[113] READDIR(3) - Linux Programmer's Manual. *readdir - Read a Directory*, September 2017. URL `https://linux.die.net/man/3/readdir`. Last accessed: August 02, 2018. [man readdir].

[114] Rid, Thomas and Buchanan, Ben. Attributing Cyber Attacks. *Journal of Strategic Studies*, 38(1-2):4–37, 2015.

[115] Riden, Jamie and Seifert, Christian. A Guide to Different Kinds of Honeypots, February 2008. URL `https://www.symantec.com/connect/articles/guide-different-kinds-honeypots`. Last accessed: August 02, 2018.

[116] Ritchie, Dennis M. The Development of the C Language. *ACM Sigplan Notices*, 28 (3):201–208, 1993.

[117] M. Roesch et al. Snort: Lightweight Intrusion Detection for Networks. In *Lisa*, volume 99, pages 229–238, 1999. URL `https://www.snort.org`. Last accessed: August 02, 2018.

[118] Sanger, David E. Obama Order Sped Up Wave of Cyberattacks Against Iran. *The New York Times*, 1(06):2012, 2012.

[119] Sastry, Anand. Honeypots for Network Security: How to Track Attackers' Activity, November 2010. URL `http://searchsecurity.techtarget.com/tip/Honeypots-for-network-security-How-to-track-attackers-activity`. Last accessed: August 02, 2018.

[120] Schneier, Bruce. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2000.

[121] Shodan. Honeypot Or Not? URL `https://honeyscore.shodan.io`. Last accessed: August 02, 2018.

[122] Spenneberg, Ralf. *Intrusion Detection und Prevention mit Snort 2 & Co: Einbrüche auf Linux-Servern erkennen und verhindern*. Pearson Deutschland GmbH, 2005.

[123] Spitzner, Lance. The Value of Honeypots, Part One: Definitions and Values of Honeypots. *Security Focus*, 2001.

[124] Spitzner, Lance. Honeytokens: The Other Honeypot, 2003.

[125] Spitzner, Lance. *Honeypots: Tracking Hackers*, volume 1. Addison-Wesley Reading, 2003.

[126] SSHD(8) - BSD System Manager's Manual. *sshd — OpenSSH SSH Daemon*, June 2017. URL `https://linux.die.net/man/8/sshd`. Last accessed: August 02, 2018. [man sshd].

[127] STDIN(3) - Linux Programmer's Manual. *stdin, stdout, stderr - Standard I/O Streams*, September 2017. URL `https://linux.die.net/man/3/stdout`. Last accessed: August 02, 2018. [man stdout].

[128] Steffens, Timo. Hacker-Jagd im Cyberspace. Grundlagen und Grenzen der Suche nach den Tätern. *c't 2017, Heft 14*, pages 122–127, 2017.

[129] Steffens, Timo. *Auf der Spur der Hacker: Wie man die Täter hinter der Computer-Spionage enttarnt*. Springer-Verlag, 2018.

[130] Stemm, Mark and Degioanni, Loris, et al. Sysdig Documentation and Wiki, September 2017. URL `https://github.com/draios/sysdig/wiki`. Latest Commit: September 22, 2017. Commit-ID: 29561418908bb8145fb522a0ebcb4bc45c0e596a. Last accessed: August 02, 2018.

[131] Stoll, Clifford. The Cuckoo's Egg: Tracing a Spy through the Maze of Computer Espionage, 1989.

[132] Stoyanov, Ruslan. Russian Financial Cybercrime: How It Works, 2015. URL `https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections/blob/master/2015/2015.11.18.Russian_financial_cybercrime_how_it_works/Kaspersky_Lab_cybercrime_underground_report_eng_v1_0.pdf`. Last accessed: August 02, 2018.

[133] strncmp(3) - Linux Programmer's Manual. *strcmp, strncmp - Compare Two Strings*, August 2015. URL `https://linux.die.net/man/3/strncmp`. Last accessed: August 02, 2018. [man strncmp].

[134] SU(1) - User Commands. *su - Run a Command with Substitute User and Group ID*, July 2014. URL `https://linux.die.net/man/1/su`. Last accessed: August 02, 2018. [man su].

[135] SYSLOG(3) - Linux Programmer's Manual. *closelog, openlog, syslog, vsyslog - Send Messages to the System Logger*, September 2017. URL `https://linux.die.net/man/3/syslog`. Last accessed: August 02, 2018. [man syslog].

[136] Sysman, Dean and Evron, Gadi and Sher, Itamar. Breaking Honeypots for Fun and Profit, 2015. URL `https://www.blackhat.com/us-15/briefings.html#breaking-honeypots-for-fun-and-profit`. Last accessed: August 02, 2018.

[137] TASS Russian News Agency (Russian Politics & Diplomacy). Kremlin Slams 'Russophobic' Allegations that Pin NotPetya Cyber Attack on Russia, February 2018. URL `http://tass.com/politics/990154`. Last accessed: August 02, 2018.

[138] TechTarget. What is Honeypot (Honey Pot)?, April 2016. URL `http://searchsecurity.techtarget.com/definition/honey-pot`. Last accessed: August 02, 2018.

[139] The Guardian. Putin: Russia Will 'Never' Extradite 13 Nationals Indicted by Mueller, 2018. URL `https://www.theguardian.com/us-news/2018/mar/04/vladimir-putin-never-extradite-13-russians-robert-mueller`. Last accessed: August 02, 2018.

[140] The Honeynet Project. The Honeynet Project, 1999. URL `https://www.honeynet.org/`. Last accessed: August 02, 2018.

[141] The jQuery Team. jQuery (JavaScript Library), August 2006. URL `https://jquery.com/`. Last accessed: August 02, 2018.

[142] The MITRE Corporation, ATT&CK. Adversary OPSEC, June 2017. URL `https://attack.mitre.org/pre-attack/index.php/Adversary_OPSEC`. Last accessed: August 02, 2018.

[143] The MITRE Corporation, ATT&CK. Group: Turla, Waterbug, WhiteBear, October 2017. URL `https://attack.mitre.org/wiki/Group/G0010`. Last accessed: August 02, 2018.

[144] The White House. Fact Sheet: President Xi Jinping's State Visit to the United States. *Press Release*, 25, 2015. URL `http://globaltraderelations.net/images/Obama.Xi_and_Obama_Fact_Sheet_Cubersecurity_9.25.15_..pdf`. Last accessed: August 02, 2018.

[145] The White House, Press Secretary (Foreign Policy). Statement from the Press Secretary, February 2018. URL `https://www.whitehouse.gov/briefings-statements/statement-press-secretary-25/`. Last accessed: August 02, 2018.

[146] The Wireshark Foundation. Wireshark · Go Deep., 1998. URL `https://www.wireshark.org`. Last accessed: August 02, 2018.

[147] Thinkst Applied Research. Canarytokens, July 2015. URL `https://www.canarytokens.org`. Last accessed: August 02, 2018.

[148] TOP(1) - User Commands. *top - Display Linux Processes*, July 2016. URL `https://linux.die.net/man/1/top`. Last accessed: August 02, 2018. [man top].

[149] TOUCH(1) - User Commands. *touch - Change File Timestamps*, December 2017. URL `https://linux.die.net/man/1/touch`. Last accessed: August 02, 2018. [man touch].

[150] UNAME(1) - User Commands. *uname - Print System Information*, February 2017. URL `https://linux.die.net/man/1/uname`. Last accessed: August 02, 2018. [man uname].

[151] United States Department of Justice. U.S. Charges Five Chinese Military Hackers For Cyber Espionage Against U.S. Corporations And A Labor Organization For Commercial Advantage, May 2014. URL `https://www.justice.gov/usao-wdpa/pr/us-charges-five-chinese-military-hackers-cyber-espionage-against-us-corporations-and`. Last accessed: August 02, 2018.

[152] United States Department of Justice. Indictment (18 U.S.C. §§ 2, 371, 1349, 1028A), February 2018. URL `https://www.justice.gov/file/1035477/download`. Last accessed: August 02, 2018.

[153] UPDATE-RC.D(8) - sysv-rc. *update-rc.d - Install and Remove System-V Style Init Script Links*, November 2005. URL `http://manpages.ubuntu.com/manpages/xenial/man8/update-rc.d.8.html`. Last accessed: August 02, 2018. [man update-rc.d].

[154] USERADD(8) - System Manager's Manual. *useradd - Create a New User or Update Default New User Information*, December 2017. URL `https://linux.die.net/man/8/useradd`. Last accessed: August 02, 2018. [man useradd].

[155] Vanhoef, Mathy and Piessens, Frank. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328. ACM, 2017.

[156] Verizon Communications, Inc. 2017 Data Breach Investigations Report - Executive Summary, 2017. URL `http://www.verizonenterprise.com/resources/reports/rp_DBIR_2017_Report_execsummary_en_xg.pdf`. Last accessed: August 02, 2018.

[157] Verma, Abhilash. Production Honeypots: An Organization's View. *SANS*, pages 1–30, 2003.

[158] Wang, Jie and Kissel, Zachary A. *Introduction to Network Security: Theory and Practice*. John Wiley & Sons, 2015.

[159] Wang, Meng. Understanding Security Flaws of IoT Protocols Through Honeypot Technologies. August 2017.

[160] Warrilow, Michael. Market Trends: x86 Server Virtualization Worldwide 2016, April 2016.

[161] Welchering, Peter. Angriff auf das Regierungsnetz - So Jagen Ermittler Hacker, March 2018. URL `https://www.zdf.de/nachrichten/heute/hack-regierungsnetz-suche-nach-urhebern-100.html`. Last accessed: August 02, 2018.

[162] WikiLeaks. Vault 7: CIA Hacking Tools Revealed, 2017. URL `https://www.wikileaks.org/ciav7p1/cms/page_14587109.html`. Last accessed: August 02, 2018.

[163] Wilkens, Andreas. Bundeshack: Russische Hackergruppe "Snake" Soll Hinter Angriff Stecken, March 2018. URL `https://www.heise.de/newsticker/meldung/Bundeshack-Russische-Hackergruppe-Snake-soll-hinter-Angriff-stecken-3984930.html`. Last accessed: August 02, 2018.

[164] Zheng, Rong and Qin, Yi and Huang, Zan and Chen, Hsinchun. Authorship Analysis in Cybercrime Investigation. In *International Conference on Intelligence and Security Informatics*, pages 59–73. Springer, 2003.