Applied Research and Innovation in Computer Science

Hide the honey from the bees

An approach to hide honey systems to insiders

Master Thesis

For attainment of the academic degree of

Master of Science in Engineering (MSc)

submitted by

Paul Lackner, BSc

1910857502

in the

University Course Applied Research and Innovation in Computer Science at St. Pölten University of Applied Sciences

The interior of this work has been composed in LaTeX.

Supervision

Advisor: Dipl.-Ing. Daniel Haslinger, BSc

Assistance: -

St. Pölten, June 29, 2021 _____ _____

(Signature author) (Signature advisor)

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.

Der Studierende/Absolvent räumt der FH St. Pölten das Recht ein, die Diplomarbeit für Lehre- und Forschungstätigkeiten zu verwenden und damit zu werben (z.B. bei der Projektevernissage, in Publikationen, auf der Homepage), wobei der Absolvent als Urheber zu nennen ist. Jegliche kommerzielle Verwertung/Nutzung bedarf einer weiteren Vereinbarung zwischen dem Studierenden/Absolventen und der FH St. Pölten.

Ort, Datum *Unterschrift*

Kurzfassung

Honeypots sind eine bekannte Methode, um neue, bisher unbekannte Informationen über einen Angreifer und Einblicke in dessen Angriffe zu erhalten. Es gibt abgeleitete Produkte, z. B.: Honeynets und Honeytoken, die als Honeysysteme bezeichnet werden. Alle diese Systeme haben gemeinsam, dass ihr ganzer Zweck darin besteht, angegriffen zu werden, der Angreifer aber nicht weiß, dass er mit einem Honeysystem kommuniziert. Honeysysteme emulieren Dienste und geben vor, ein echter Dienst zu sein, um einen Angreifer dazu zu verleiten, mit ihnen zu kommunizieren und so (neue) Angriffsmethoden kennen zu lernen. Honeypots sind ein einzelner Host oder Dienst, der Angreifer anlockt. Honeynets sind mehrere Honeypots in einem Netzwerk. Diese Netzwerke können rein fiktiv sein oder sie können Produktionsnetzwerke klonen, um authentischer zu sein. Honeywalls sind die Border-Gateways, die den Datenverkehr wie Firewalls filtern und ihn entweder an das Produktionsnetzwerk oder das Honeynet weiterleiten. Honeytokens sind ein wenig anders, da sie selbst keine Rechenleistung haben. Sie sind lediglich Daten in Form von Dateien, Einträgen innerhalb von Dateien oder Datenbankeinträgen. Honeytokens können einen bereits erfolgten Einbruch signalisieren, wenn mehrere Einträge in Dateien außerhalb der Unternehmensdaten gefunden werden (z.B.: fiktiver JFK-Eintrag in Krankenhausdateien). Da Honeysysteme am besten funktionieren, wenn der Angreifer viel Zeit mit dem System verbringt, ist es notwendig, einen Dienst gut zu emulieren und ein Honeysystem so gut wie möglich zu tarnen. Angreifer sind sich jedoch über Honeysysteme im Klaren und wollen ihre Zeit nicht mit einem unrentablen Dienst verschwenden, und sie wollen auch nicht, dass ihre Techniken aufgedeckt werden. Aus diesem Grund wurden mehrere Techniken zur Erkennung von Honeysystemen entwickelt, die wiederum zu Techniken führten, die Honeysysteme noch besser tarnen.

Wie bereits erwähnt, wurden viele Arbeiten über Honeysysteme veröffentlicht, auch über das Verstecken derselben. Aus offensichtlichen Gründen sind diese Forschungen noch nur sehr spärlich veröffentlicht worden. Die einzige relevante Veröffentlichung stammt aus dem Jahr 2003 und ist daher stark veraltet. Firmen, die ein solches Papier veröffentlichen würden, implementieren wahrscheinlich die in diesem Papier erläuterten Funktionen und würden damit ihr eigenes Konzept sprengen. Dieses Konzept könnte die Sicherheit gegen Insider-Eindringlinge und Spionage erhöhen. Die Ergebnisse können einen Paradigmenwechsel bewirken, da sich die Organisationen wirklich darauf konzentrieren können, abtrünnige Mitarbeiter zu finden, da “Nur Angreifer auf einen Honeypot zugreifen; normale Benutzer haben nicht die Absicht, ihn zu be-

nutzen". Das bedeutet weniger Überwachungskosten für die Organisation und mehr Privatsphäre für die Mitarbeiter. Da Honeysysteme nicht von normalen Anwendern, sondern nur von Angreifern genutzt werden, sind keine Datenschutzklagen von Angreifern zu befürchten.

- Wie kann man ein Honeysystem nach innen verbergen?
- Welche Honeysysteme können nach innen versteckt werden?
- Vor welchem Personal kann ein Honeysystem versteckt werden?
- Ist es sinnvoll, ein Honeysystem nach innen zu verstecken?

Die Hypothese ist, dass alle Honeysysteme für alle Mitarbeiter in einem Unternehmen versteckt werden können, außer für die, die dafür zuständig sind.

Der Ansatz zur Beantwortung dieser Fragen ist, wirklich einen Honeypot in einer Firma zu installieren und zu versuchen, alle Mitarbeiter zu umgehen. Bei der Enthüllung wird der Mitarbeiter in die Materie eingeweiht und der Fehler wird dokumentiert. Die Mitarbeiter werden hinsichtlich ihrer Kenntnisse der (IT)-Infrastruktur in Gruppen eingeteilt und separat beobachtet, ob und wann sie den Honeypot entdecken. Um ein gutes Ergebnis zu erzielen, muss wahrscheinlich der gesamte Einsatzprozess eines Honeysystems geändert werden. Da diese Honeysysteme nicht auf Angreifer von außen, sondern auf Insider abzielen, ergeben sich einige weitere Fragen zum Prozess. Diese Herausforderungen werden während der Forschung auftauchen und müssen während des Projekts gelöst werden, was das Problem komplex und schwierig macht.

Diese Arbeit erstellt eine Bewertung der Mitarbeitergruppen und Assets, um Erkenntnisse über deren Beteiligung an einem solchen Projekt zu gewinnen. Sie stellt auch ein Experiment vor, wie diese formulierten Hypothesen auf Organisationen in der realen Welt anwendbar sind und beantwortet die Fragen, wie man welches Honeysystem vor wem verstecken kann und ob es sinnvoll ist, dies zu tun.

Abstract

Honeypots are a well known method to gain new, previously unknown information about an attacker and visibility about their attacks. There exist derived products, e.g.: honeynets and honeytoken, which will be described as honey systems. All these systems have in common that their whole purpose is to be attacked, but to leave the attacker unaware of his communicating with a honey system. Honey systems emulate services and pretend to be a real service to lure an attacker to communicate with it and therefore to learn about (new) attacking methods. Honeypots are a single host or service luring attackers. Honeynets are multiple honeypots in a network. These networks could be purely fictious or they can clone production networks to be more authentic. Honeywalls are the border gateways, filtering traffic like firewalls, and routing the traffic either to the production network or the honeynet. Honeytokens are a little bit different, as they have no calculation power itself. They are just data in form of files, entries within files or database entries. Honeytokens can signal an already happened intrusion, when several entries in files are found outside of company data (e.g.: fictious JFK entry in hospital files). As honey systems work best if the attacker spends a lot of time with the system, it is necessary to emulate a service good and disguise a honey system as good as possible. However, attackers are aware of the possible existence of honey systems and do not want to waste their time with a non-profitable service and they do not want their techniques getting revealed, which is why multiple honey system detection techniques are developed, which lead to newer techniques for honey systems to better avoid detection.

As already mentioned, a lot of papers have been published regarding honey systems, also about hiding them. This research about how to implement them in a concrete surrounding still has never been published because of obvious reasons. The only relevant publication dates to the year 2003 and is therefore highly outdated. Companies publishing such a paper would thereby probably bust their own concept of implementing these features. This concept could increase security against insider intrusion and espionage. The results can shift paradigms as the organisations can really focus on finding rogue personnel, as “Only attackers access a honeypot; normal users have no intention of using it”. This means less surveillance costs for organisation and more privacy for employees. As honey systems are not used by normal users but only by attackers, no privacy concerns are to appear.

Honey system primarily address strangers, attackers from outside of the company. These can cause severe

damage to a company, but insiders have a better plan of a company and how to disguise security patterns. Honey systems are not publicly used as monitoring system to protect against internal attacks. As insiders usually know the company structure it is difficult to implement a shadow system, only selected personnel know about. A shadow system can furthermore attract attention of an inside intruder, which makes honey system valuable as they only address attackers. The research questions therefore are:

- How to hide a honey system to the inside?
- Which honey systems can be hidden to the inside?
- From which staff can a honey system be hidden?
- Does it make sense to hide a honey system to the inside?

The hypothesis is that all honey systems can be hidden to all staff in a company, except to the ones being in charge of it.

The approach to answer these questions is to really install a honeypot in a company and try to circumvent all of staff. When being disclosed, the staff member will be introduced to the matter and the error will be documented. Staff will be divided into groups regarding their knowledge of the (IT)-infrastructure and separately watched if and when they detect the honeypot. To achieve a good result, the whole deployment process of a honey system probably need to change. As these honey systems do not target outside attackers but insiders, several further questions about the process appear. These challenges will appear during the research and needs to be solved during the project which makes the problem complex and difficult.

This work creates an evaluation of the staff groups and assets to provide knowledge about their involvement in such a project. It also presents an experiment on how these formulated hypothesis apply to real world organisations and answers the questions on how to hide which honey system from whom and does it make sense to do so.

Contents

1 Introduction

The consecutive race between a red team and a blue team, therefore between attackers and defenders in terms of computer security won't settle down in the next years. The attackers will always be ahead of the defenders when it comes to exploit systems. To minimize the advance of the attackers many technical and organisational improvements have been made in the last years. Organisational methods like a software security lifecycle and better programming languages improved security of software applications and operating system (OS) in general, vulnerabilities found by testing methods and security researchers are fastly patched, and public exploits and coherent mitigation are notified. To detect exploits and attack techniques, honey systems, especially honeypots are widely used in a research and productive context[1]. Honeypots enable defenders to create profiles about attackers[2]:

- where they come from
- their technological level
- how they access a device
- what they are interested in
- how well the existing security already works

Honey systems have an efficient detection method, so only a few are needed to get a large dataset of attacks[3], therefore most of the security sales companies operate a honeynet[4] to generate security patches for the public while local companies operate their own honeypots to detect targeted attacks[5].

Honeypots are no new thing [1], but still relevant as they are the first line of defense to detect attack techniques and a reliable instrument to detect intruders. Honey systems vary from simple to complex implementations and can grant attackers a variety of interaction levels while defenders get a variety of insights

[1] https://www.businesswire.com/news/home/20190516005157/en/

[2] https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot

[3] https://www.techrepublic.com/article/kaspersky-honeypots-find-105-million-attacks-on-iot-devices-in-first-half-of-2019/

[4] https://dtag-dev-sec.github.io/mediator/feature/2018/02/23/potherder.html

[5] https://cybernews.com/security/honeypots-how-security-teams-use-bait-to-protect-against-cyber-threats/

depending on the interaction levels. Honeypots may be standalone systems as well as implemented into existing networks. This variety of possibilities and the low entry level for administrators and researcher as well as the effectiveness of honey systems grants the high popularity.

As common defense techniques apply good against common attackers, insider or attackers already being persistent in your network are not in a common security scope[6]. This work aims to increase security against insider threats or attackers already operating from the inside via already established honey system concepts. These concepts although have only targeted external attackers. Therefore four major research questions are stated:

1. How to hide a honey system to the inside?
2. Which honey systems can be hidden to the inside?
3. From which staff can a honey system be hidden?
4. Does it make sense to hide a honey system to the inside?

To answer these questions, some other non-obvious, but related questions arise, which need to be answered in order to answer the major questions:

1. What to do when being found by an employee?
2. Which people need to be involved?
3. How to maintain the system?
4. How to physically hide the system? Will it be a VM on a hypervisor?
    a) Can already existing devices be used (e.g.: climate sensors)?
    b) Can already existing processes be used to disguise the real purpose of a device?
    c) Can a device be placed in a rack of another department (so nobody feels in charge of it)?
5. How and where to log?
6. How to hide wanted traffic (management, logging, etc.)?
7. How to hide malicious traffic (If honeypot traffic is not visible to security staff)?
8. How and where to document all of that?

Answering these questions will give an understanding of the underlying problems and will bring research nearer to increase security against these attackers.

[6] https://www.dni.gov/files/NCSC/documents/news/20210319-Insider-Threat-Mitigation-for-US-Critical-Infrastru-March-2021.pdf

1.1 Thesis Outline

This thesis is organised in several parts which will be shortly described now.
In chapter 1 the topic, problems, challenges and motivation are introduced as already read.
Prerequisites and fundamental knowledge are described in chapter 2. It defines the terminology of various honey systems, their properties and use cases and introduces some implementation examples. It concludes with a short overview about some legal questions and an introduction to a commonly used threat information exchange.
In chapter 3 the related work is listened and shortly summarised.
The theoretical approach and some thoughts about the research questions are made in chapter 4. It describes the concept of hiding or disguising a honey system to the inside and examines it from different points of view.
An experiment is described and discussed in chapter 5. The experiment implements one of the concepts of chapter 4 and examines its applicability to the wild.
Last but not least chapter 6 concludes this work and the research questions. It also provides recommendations for implementations as well as further research that needs to be done.

2 Prerequisites

To basically understand the purpose of a honey system, the following paragraphs may be helpful. They explain the basic terminology, purposes and implementation methods of honey systems. The last paragraphs explain some legal aspects, as well as a threat exchange system. The following sections (2.*) marked with an asterisk are taken from a previously published work of myself [2].

2.1 Terminology of Honeysystems*

Honeypots, honeynets, honeywalls, and honeytoken all serve the same purpose, namely to detect intruders and analyse their intrusive behaviour. No legitimate user would ever access a honey system [3], even a connection attempt is considered an attack [4].

2.1.1 Honeypot

Invented in the 1990s, a *honeypot* is the best known application of the honey systems [1]. It is a single host, network device or daemon [4] luring potential attackers to distract them from valuable network ressources [5]. “A honeypot is a security resource whose value lies in being probed, attacked, or compromised.” [6]. “A honeypot is a resource which pretends to be a real target.” [4]. A honeypot itself is not a security fix but helps fixing issues by gaining information about attacks. “The main goals are the distraction of an attacker and the gain of information about an attack and the attacker” [7].

2.1.2 Honeynet

A *honeynet* is a network of multiple honeypots [8], [9]. Analogous to honeypots, the entire network is to be attacked. The honeynet is not emulated and therefore can be a copy of the production system [10]. “Honeynets represent the extreme of research honeypots. They are high interaction honeypots, which allow learning a great deal; however they also have the highest level of risk. Their primary value lies in research and gaining information on threats that exist in the Internet community today. Little or no modifications

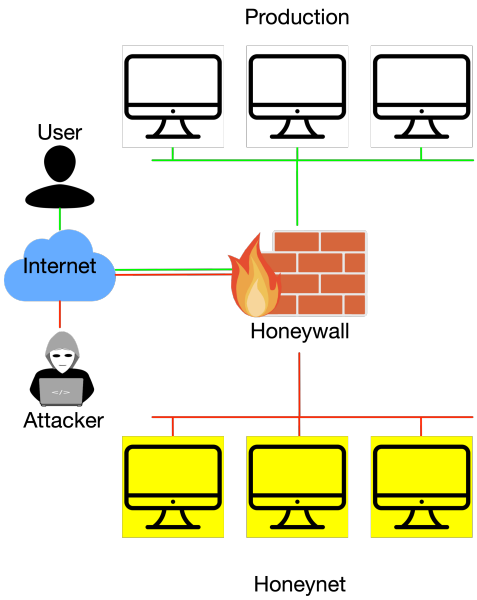

Figure 2.1: A honeynet, which consists of honeypots, duplicating the production system and being separated by a honeywall. The honeywall routes legimite users to the production network (white hosts) and attackers to the honeynet (yellow hosts).

are made to the honeypots of the honeynet to provide a plausible copy of the production net. This gives the attackers a full range of systems, applications, and functionality to attack. From this it can be learnt a great deal, not only their tools and tactics, but also their methods of communication, group organization, and motives" [8]. Low and high interaction honeypots are further explained in section 2.2. Figure 2.1 shows an example of a honeynet.

2.1.3 Honeywall

As also visualised in Figure 2.1, a *honeywall* is the perimeter border between honeynets and productive systems, although the term honeywall is not clearly defined. Some literature describe it as a gateway [10], some describe it as a layer-2 or layer-3 filtering bridge firewall/Network Intrusion Detection System (NIDS) [11]. It is also not clear whether the term "honeywall" refers to the product name of *The Honeynet Project* or the basic functioning in honey environments itself. The product *honeywall* is based on *iptables* rules, *Snort*, and *Snort-inline* [11]. Other implementation use similar tools. In this paper, the term honeywall refers to a general implementation. Honeywalls support interception of SSL connections and decide if incoming traffic is malicious and therefore needs to be redirected to a honeynet, or if it is valid and therefore redirected to the productive system.

2.1.4 Honeytoken

Unlike honeypots or honeynets, *honeytokens* are just data in the form of files, entries within files, or special strings [10] [12]. The data looks valid even though it is fictional and does not have any production use. The files are monitored in case of their modification. Data and strings are monitored as well, e.g. via *Google Alerts*. Google Alerts, a web service, notifies a user if a string appears in Googles search. Therefore, when a user is notified about a published honeytoken, it is likely that this honeytoken has been stolen and a successful intrusion occurred. A paper defines the following honeytoken properties [13]:

- Believable: A honeytoken looks like valid data.
- Enticing: A honeytoken lures an attacker.
- Conspicuous: A honeytoken is easily found.
- Detectable: Interacting with a honeytoken generates an alert.
- Variability: Various honeytoken do not contain the same information that create a connection between them.
- Non-interference: A Honeytoken does not interfere with desired data or system interactions.
- Differentiable: A legitimate user can differentiate between a honeytoken and actual data while an intruder cannot.

2.2 Properties of Honeypots*

Another survey [14] defines honeypots as devices which distract attackers from valuable machines, provide early warnings about attacks and allow in-depth examination of adversaries during and after the exploitation. To this end, it is in the interest of a defender that an attacker interacts with the honeypot over an extended period of time, while being closely monitored. There are several types [6] and several use cases [14] of honeypots. While each type can be used in each use case, there are different advantages and disadvantages for each honeypot configuration.

2.2.1 Types

A *low interaction honeypot* has a very limited set of commands available for an attacker. While there is not much information one can obtain about attackers, the risk of damaged production systems due to a successful attack through that honeypot is also low [15].

A *high interaction honeypot* has the goal to obtain a maximum amount of information about the attacker. The honeypot allows itself to be used, tampered with, or even be damaged. High interaction honeypots often

are a clone of a production server. The goal is mainly to learn about novel attack techniques [15]. When deploying a high interaction honeypot, considerations about an extremely resilient monitoring system have to be made, to obtain authentic information of a partially compromised honeypot.

A *medium interaction honeypot* is in between of a low and a high interaction honeypot. While a low interaction honeypot only provides a limited set of commands and a high interaction honeypot grants access to the OS of the honeypot, a medium interaction honeypot has a significant higher amount of available commands and interactions but does not grant any access or interaction with the OS directly. Medium interaction honeypots emulates a service [14].

2.2.2 Use Cases

Widely known literature regards two use cases; research and production. This paper introduces a third use case: vulnerability scan.

The idea of a *research* honeypot is to provoke an attack and learn about the attacker. The purpose of a research honeypot is to learn about new techniques and tools of attackers. It also serves to learn about new combinations of tools attackers use to tamper with systems. There is no intention to defend the system against attackers [15].

A *production* honeypot is only used in productive networks to obtain information about attackers invading the productive network. The purpose is not to gain maximum intel about the attacker but to detect the interest of the attacker. Spitzner and Schneider defined three interest groups within security issues: Detection, Prevention and Response [15]. A honeypot can provide value to Detection and Response.

HosTaGe constitutes a further use case, the non intrusive *vulnerability scan* [16]. *HosTaGe* is a honeypot for mobile phones which scans public wireless networks. It detects malware spreading from devices within the same network and checks the basic security of public networks. It is considered to be a honeypot-to-go. *HosTaGe* also supports Industrial control system (ICS) protocols, making it relevant for industrial companies [17].

When implementing a honeypot, several problems should be considered [14]:

- Data types: Data provided should look authentic to gain the attention of an attacker but it should not be possible to harm the company with this data.
- Uplink liability: Honeypots can and will get compromised, which enables an attacker to attack other systems with it. Considerations about placing a honeypot in separate networks, maybe even a different public IP range than the production network of the enterprise have to be made (see more in section 2.5).

- Build your own honeypot?: Most of the honeypot frameworks are open source and therefore customisable. The established honeypots professionally emulate the services they offer, when configured correctly. Additional services have to be programmed by the respective end-users themselves.
- Hiding: A honeypot should not obviously appear as a honeypot (see subsection 2.4.3).

Some researcher created a method to evaluate the potency of a honeynet which can be deployed to any honeynet [18].

2.2.3 Advantages and Disadvantages of Honeypots

Honeypots have particular advantages over traditional NIDS and network security approaches [14]:

- Small data sets: Only traffic addressing the honeypot is being observed. This can also be a disadvantage however.
- Minimal resources: Only attackers access a honeypot; normal users have no intention of using it. There is not always the need to use state-of-the-art hardware as the system does not have to carry normal production traffic and interactions.
- Simplicity: Honeypots, especially low interaction honeypots, are “simple and flexible”. They feature easy deployment and update routines as they do not offer the whole functionality of the original service.
- Discovery: Honeypots can help discover new tactics and tools directed at them.

Naturally, honeypots also have some disadvantages. To decide if a honeypot is the right device to monitor one’s network, one has to consider the following points [14]:

- Limited Vision: Only the traffic that hits the target is visible and analysed. Traffic that does not target the honeypot in any way will not be detected. This can also be an advantage.
- Discovery and Fingerprinting: Honeypots can either emulate a service or provide the service itself. All services have different fingerprints in terms of reaction time and properties when using different network stacks, operating systems, and hardware. Nevertheless some fingerprints can reveal the implementation of a service. If a service is emulated, the fingerprint can reveal the emulating service (e.g.: honeypot tool). It might be possible to distinguish a real service from a honeypot from network meta data.
- Risk of takeover: A honeypot may be used as an attacking device if it is taken over. So a honeypot should be monitored to determine, if it is still working as intended or already got compromised.

2.3 Implementation Examples of various Honey Systems*

Honey systems are used in various ways with various goals to achieve. The following subsections enumerate some examples of the possibilities.

2.3.1 Honeynets as a NIDS

A Network Intrusion Detection System is a system which detects known attacks [19] and warns administrators mostly through a Security Information and Event Management (SIEM) tool about incidents. During a project a honey net was created through container orchestration and *Conpot* as a honeypot backend [20]. They define three modules to achieve an easy-to-work-with honeynet: the deployment module deploys a honeypot as a container with the desired properties. The management module manages the deployed container in terms of power management and honeypot properties. The intrusion detection module is the brain of this system. It processes traffic provided by the honeypots, trying to find attack patterns. Using a Support-Vector-Machine (SVM) algorithm the traffic data, especially the meta data (connection duration, protocol type, number of source bytes, number of destination bytes, whether it comes from the same host, the number of wrong segments, the number of urgent packets, etc.) is used to train a model, which was initally trained with the KDDCUP99[1] data set (a data set from 1999). The intrusion detection module uses this model to create alerts. They argue, since every connection to a honeypot is an attack per definition [4], this is a way to determine whether an attack is just random noise or a targeted attack against the system and to achieve a better detection rate with a lower amount of false positives [20]. The detection rate of 89% still remains low compared to common detection rates using the SVM algorithm. A newer approach describes a similar concept [21].
A different approach describes a honeynet out of high interaction honeypots to detect a botnet in its creation phase using machine learning to detect hidden patterns [22]. Furthermore, honeypots can organize themselves with a (private) blockchain to act and save data distributed and be more resistant to takedowns [23].

2.3.2 Honeywall

As seen in Figure 2.1, a honeywall is the perimeter border between a honeynet, a possible production net, and the internet. Its three main goals are: data capture, data control, and automated alerting. This is achieved by combining a layered firewall with NIDS/IPS and a monitoring tool. The honeywall decides if traffic is malicious or valid, and helps to correlate traffic from or to various honeypots in the honeynet [24]. A problem

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

is the generation of good training material. To generate malware traffic signatures which let the honeywall decide if traffic is valid or malicious, malware traffic needs to be routed to the honeynet by the honeywall. Obviously, this is a chicken-and-egg problem, which can be mitigated by reversing the function of a NIDS. While a NIDS usually works by block-listing malicious traffic [19] it can also pass-lists legitimate traffic: All known legitimate traffic is sent to the productive network, everything else to the honeynet, where traffic is used to generate detection patterns for the honeywall.

2.3.3 Honeytokens as an Active Defense

Honeytokens could be used as an active defense [3]. An explanation of this statement necessitates an understanding of the concepted different zones, visualised in Figure 2.2.

Zone 0 acts as the perimeter zone and may contain a firewall (honeywall), NIDS, and Web Application Firewall (WAF). Honeytokens are not placed there, because they are meant to address attackers who are able to circumvent these defense technologies. Although, on the perimeter border, an IDS rule can detect the usage of a honeytoken [25].

Zone 2 represents the web application. Honeytokens are placed here as they enable the application to react just-in-time to an attack. A URL parameter (*https://example.com/site.php?admin=false*) is an example of how applications react to the usage of honeytokens. When accessing the site with the parameter *admin=true*, the application knows to call a specific function defending against an attack. A further implementation is a web application honeypot to track bot crawls on a website [26].

Zone 1 and 3 represent the web server and possible databases. These zones contain classic file or database-entry tokens to collect as much data as possible about the attacker if zone 2 is compromised. A webserver for example, storing honeytokens as files, can monitor them through the meta data. An exfiltration of these honeytokens is noticed by monitoring the file metadata, or by creating a trigger for the usage of a query for these honeytoken database entries.

The higher the zone number, the further an attacker has advanced into a system.

An example of the usage of honey token and the reaction of webservers follows: [3]:

- Session manipulation: If an attacker accesses a honeytoken, the session in use may be terminated or isolated into a sandbox mode. Also, every or randomly chosen requests may be answered with a HTTP 200 OK status code to feed the attacker with false information. A request rate limiting or intentional connection timeouts may further slow down the attacker.
- Generated vulnerabilities: Once the system detects attackers it dynamically generates vulnerabilities to “keep the attackers happy” and to gain more time to study the attackers.

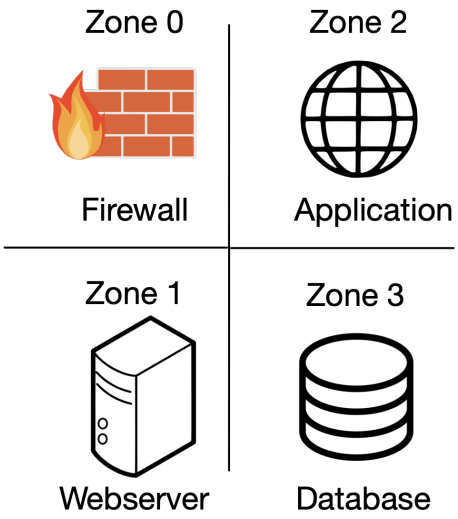

Figure 2.2: An overview of the zone concept of Petrunić in terms of honeytoken defense. Zone 0 represents the perimeter border where no honeytoken is placed. Zone 1 is the web server which hosts the web application. Zone 2 portrays the web application, which reacts directly to an attacker when using a honeytoken. Zone 3 is the database. The database hosts honeytokens as database entries. It can trigger actions when a specific entry is accessed [3].

- Attack the attacker: The generated vulnerabilities can deface attackers and reveal their identity by forcing them to establish a direct communication channel [27] [3].
- *robots.txt*: Fake URLs and entries in the *robots.txt* help to identify attackers and malicious web spiders. *robots.txt* defines whether a web spider is allowed to index an URL or is not. When one of the fake URLs in *robots.txt* is accessed by either an attacker or malicious web spider, a specific reaction may get triggered.

Honeytokens are either handcrafted or generated through a tool like *HoneyGen* [28]. Like honeypots, honeytokens slow down attackers [29], and feed them with false information[2].

2.3.4 Real world implementations

Honey systems are already used by different companies with different implementations. While companies usually do not publish their use of honey systems, some are well known to use them and even openly admit to use honey systems. Nevertheless, enterprises do not publish the exact implementation and version of their honey system.

[2] https://arstechnica.com/information-technology/2017/05/macron-campaign-team-used-honeypot-accounts-to-fake-out-fancy-bear/

Many telecom providers, as well as security enterprises (firewall/NIDS manufacturers, antivirus/HIDS manufacturers, etc.) sustain many honeypots in a honeynet. They are distributed worldwide and create threat information and an attack map, among other things. Associations hide their critical infrastructure with honey systems and improve security efforts with the obtained information. Also, hospitals and medical facilities test the integrity of their personnel with a bogus patient record, for example a honeytoken called "John F. Kennedy" [30]. Table 2.1 shows a comparison of various honeypots.

2.4 Mock the Bear, Manipulate the Honey*

When operating any honey system there is always the risk of a compromised system as well as a defaced honey system. Either way, a honey system should work as intended. Monitoring and logging tools must be resistant to attacks to still record trustworthy information about an attack and an attacker. Also, attackers are capable to detect honey systems with various methods. This demands various hiding methods. The following paragraphs explain monitoring and logging methods, ways to distinguish honey systems from productive systems, and methods to mitigate a detection and to hide honey systems.

2.4.1 Find the Bear, Monitor the Honey

Various ways of monitoring honeypots exists. When honeypots became popular, *SEBEK* was a popular monitoring tool, superseded by *XEBEK*, a *SEBEK*-like tool optimised for virtual machines. These are data/traffic capture kernel modules which send data to a central logging server to monitor and visualise events [31]. Nowadays, standard tools like *syslog* and Trusted Automated eXchange of Indicator Information (TAXII) in combination with a central monitoring server are more common [32]. Since a honeypot is accessed by an attacker, there has to be a focus on log authenticity. Sending logs immediately to a remote logging station and not storing them locally on a compromised honeypot is part of a good log handling.

2.4.2 Detect the Honey

A recent paper describes a method to monitor and analyse botnets with honeynets [34]. To get authentic results, considerations must be made on the fact, that many different ways of detecting honeypots have been developed by attackers to avoid honeypots. A quite old paper describes a way to detect honeypots in a botnet [33]. The underlying problem is still present: "Should a honeypot attack other systems to look authentic?" A botnet is a network of infected machines controlled by a bot controller [35]. A bot controller commands a new bot to attack a list of hosts. This list of hosts or addresses include sensors, controlled by the

Honeypot	Host	Service	Description
ADBHoney	Linux	ADB	emulates an Android Debug Bridge
Conpot	Linux	ICS/Supervisory Control and Data Acquisition (SCADA)	emulates different ics protocols like modbus
Cowrie	Linux	ssh, telnet	against brute-force attacks
dionaea	Linux	shellcodes	successor of *Nepenthes*
GhostUSB	Windows	USB storage	to detect malware trying to spread over USB drives
Glutton	Debian	any network port	answers to any network communication
Heralding	Linux	ftp, telnet, ssh, rdp, http(s), pop3(s), imap(s), smtp, vnc, postgresql, socks5	gathers credentials
HoneyD	Linux	*	modular honeypots; one of the oldest honeypot implemetations; no longer developed
HoneyDoc	python	Honeytoken	creates fake documents and tracks who opens them
Honeytrap	Debian & MacOS	*	decentralized honeypot framework to create honeynets
HosTaGe	Android	any network port	vulnerability scan of public wireless network
KFSensor	Windows	any network port	proprietary enterprise honeypot
mailoney	Linux	SMTP	Detects attacks on SMTP services
medpot	Linux	HL7, FHIR	honeypot for medical protocols (Health Level 7, Fast Healthcare Interoperability Resources)
RDPY	Linux	RDP	RDP implementation in python
SNARE & TANNER	Linux	http(s)	SNARE acts as network agent, TANNER as analysis and classification tool
T-Pot	Linux	*	T-Pot is a collection of various honeypots and put together

Table 2.1: Comparison of various honeypot implementations.

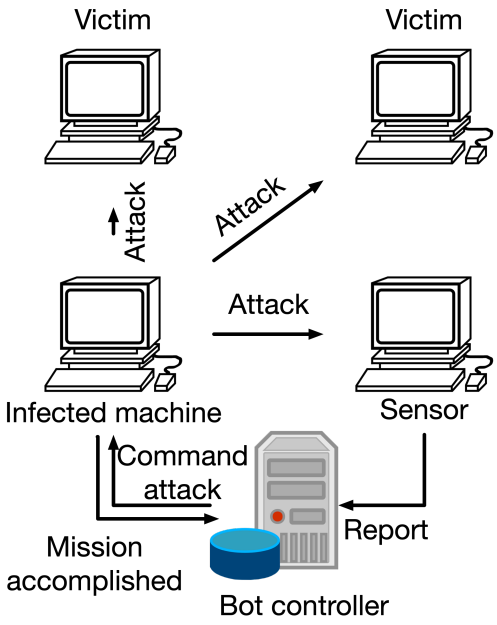

Figure 2.3: Visualisation of a bot controller trying to verify if the infected host is a normal infected machine or an infected honeypot. The bot controller commands the infected machine (bot) to attack a list of hosts including a sensor. The bot attacks the given list of computers and sends a "Mission accomplished" to the bot controller. A honeypot would not attack other hosts but reports a "Mission accomplished" to the bot controller. The sensor reports to the bot controller that it got attacked by that specific host. The bot controller correlates reports of the sensor with his list of probably infected machines and verifies the infection and is thus able to identify honeypots [33].

attacker, which report every attempt to contact them to the bot controller. When the bot controller receives an IP address, to which the controller commanded an attack before, it is assumed a valid bot, otherwise it might be a honeypot trying to sanitize its outgoing traffic to not attack others [36]. This introduces the dilemma of sanitising outgoing traffic due to legal affairs and getting authentic results. For more information on why honeypot operators sanitize outgoing traffic, see section 2.5. Figure 2.3 shows this scenario.

2.4.3 Hide the Honey

Honeypots, honeynets, honeywalls, and honeytokens need to be concealed from the attackers. The first step to hide them is to determine the type of attacker they are hidden from. Usually, "attacker" means everything outside of an organisation or company. To hide a honeypot from the outside, it should behave like the emulated service in terms of user interaction. This means that the front-end and user responses, especially the reaction time of the service, remain the same as the original service [37]. The back-end can be adjusted to the operators' needs. In the best case, an attacker does not know about the structural design of an organisation, so little to no management effort is required to hide honey systems.
The emulation of several things, including the network stack and processing must be authentic to the real service to disguise the honeypot [37]. Network scanning tools like *nmap*[3] look for protocol quirks to identify the exact implementation. Depending on the purpose, honeypots should or should not emulate the exact quirks. Various methods to properly emulate services are suggested [25]:

- Vulnerability emulation: Only the vulnerable part of a service or OS is emulated. *Nepenthes* [38] and *PHP.HoP* [25] uses this technique. This obviously works only for already known vulnerabilities!
- Connection tarpitting: Tarpitting in terms of honeypots means delaying network traffic and service responses. Network traffic may be delayed by manipulation of the window flag in Transmission Control Protocol (TCP/IP) packets. Service responses may be delayed deliberately by waiting milliseconds before sending them. Appropriate waiting times might help to make a honeypot more authentic; e.g. sending a command to a machine via a controller implies waiting for a response from a controller which waits for the response of the machine, which takes time. Immediately sending a response likely unmasks a honeypot.
- Traffic redirection: Suspicious traffic gets redirected to a honeypot. For example, a connection establishment to an unused IP or suspicious traffic, detected with help of a NIDS, can be redirected to a honeypot. However, a NIDS only detects already known attack vectors and needs to know the network topology or at least the used IP addresses to detect unused IP addresses.

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

The techniques described here hide a honey system from an external attacker. Hiding honey systems from personnel or specific groups of personnel of an organisation or company was not found in literature.

2.5 Legal Affairs and Privacy*

When operating a honeypot one has to consider several legal aspects.

Liability is an important factor to consider. Activity and traffic from the honeypot is linked to the organisation hosting the honeypot. If a honeypot is compromised and attacks other systems or acts as an illegal market place for example, there are legal consequences for the organisation. Especially neglected honeypots, which are not monitored regularly, are a popular target. To prevent consequences, aside from the technical aspects, the goal and methods of a honeypot should be documented in detail [14].

The right to monitor is competitive to the right of *privacy*. As this tension is part of the daily life of an IT-security engineer, this applies to honey systems as well. Most countries already passed laws regarding privacy matters, such as the Fourth Amendment, the Wiretap Act, the Patriot Act, and the Freedom Act in the USA and the General Data Protection Regulation (GDPR) in the EU. A decision, whether and how information can be logged, is made regarding these laws [14]. A profound analysis of privacy matters, concerning honey systems, has already been done based on EU laws, already including GDPR, in 2017 [39].

2.6 Threat Information Exchange

All these different systems result in a variety of logging formats. To standardize the logging, to be able to combine logs of different honeypots and other network components, and to analyze and visualize them, one of the innovations was the specification of Structured Threat Information eXpression (STIX) and TAXII. The most common logging method used, apart from STIX and TAXII, is *syslog*.

2.6.1 STIX

STIX [40] is a language to standardize the specification, capture, characterisation, and communication of cyber threat information. STIX defines several different object domains (threat actor, incident, exploit, campaign, indicator, etc.) and relationship roles (relationship, sighting) which describe the relationship between various object domains and their frequency of occurrence.

2.6.2 TAXII

TAXII [41] is a standardized set of services and messages to exchange threat information. TAXII defines different network components, roles and (Consumer, Producer) concepts as well as various sharing models between peers [42]:

- TAXII Data Feed is a collection of threat information in STIX format.
- TAXII Message is a request or response of one entity to another.
- TAXII Message Exchange is a collection of requests and responses between two parties to a specific subject (comparable to an email conversation).
- TAXII Service is a functionality hosted by an entity that can be accessed from another entity.
- Hub and spoke: A hub officiates as the center and handles requests from spokes (peers). It also updates all other peers. Communication occurs only between a spoke and the hub.
- Source and subscriber: A source is the center and broadcasts all information to the subscribers. Connections from the subscriber to the source are not allowed.
- Peer to peer: Every peer in the system can communicate with every other peer. Information flows in both directions.

TAXII is designed to support STIX. In contrast to *syslog*, which transmits in clear text by default, TAXII uses a transport encryption protocol (HTTPS) to transmit information between two peers [43].

3 Related Work

This chapter represents the state of the art, what has already been done.

Cheswick described the first honeypot already in the early 90s [1].

HoneyD was the first open source honeypot, which was presented in a paper by its author Provos [4]. It already enabled to customise the installation via plugins. HoneyD is no longer under active development.

Petrunić created a method to use mostly passively used honeytokens as an active defense method against attackers and attack attackers [3].

Pouget *et al.* defined a clear and simple general understanding of the most known honey systems and created a well known terminology [5]. This is an important work to organise international research.

Spitzner is a well known expert about honeypots. He created a lot of work about this topic, as well as a comprehensive text book about honeypots [6] and the first work about catching an insider with honey systems [10].

Baumann *et al.* created a first extensive master thesis about honeypots [7].

Project is a non-profit organisation which creates knowledge and software on various honey systems. They are the organisation who created the product honeywall. They created a series of articles called “Know your enemy” which had a lot of impact in honey system development [8] [9].

Bowen *et al.* made a first notable research about baiting inside attackers with honeytokens [13].

Mokube *et al.* did a small survey about honeypots, that provides a good first insight into this topic [14].

Mairh *et al.* created an extended survey about honeypots [15].

Vasilomanolakis *et al.* created a newly honeypot use case with their research implementation “Hostage” [16], the public network vulnerability scan. It also got an update with slightly different authors [17].

Ren *et al.* created a theoretical approach to evaluate a honeypot potency in the network [18].

Wang *et al.* did a research about how to extend honey system accuracy with machine learning [20]. They added a SVM algorithm to a honeypot and were able to achieve a 89% accuracy in malicious traffic detection. This concept has already been updated [21].

Martínez Garre *et al.* created a honeynet to detect the creating of a botnet [22].

Shi *et al.* created an approach to distribute honeypot data in a blockchain to be more resistant against take-downs and to act with a profound data base [23].

Qassrawi *et al.* created an overview of deception methods of honeypots to gather as much information as possible during attacks [25].

Lewandowski *et al.* created an approach to analyse web bot on websites [26]. They created a website as a honeypot to detect how web spiders crawl through a website.

Djanali *et al.* created a similar approach to Petrunić with his active honeytokens [3], but they created a honeypot to achieve a counter attack to attackers and reveal their identity [27].

Bercovitch *et al.* build an automated honeytoken generator that is able to build a honeytoken for every environment and the token not be detectable by humans [28].

Sandescu *et al.* used honeytokens to help understanding an attack and to influence the execution of it [29].

Spitzner created an early comprehensive general understanding of honeytokens [30].

Quynh *et al.* described XEBEK and SEBEK, honeypot monitoring tools [31].

Wagner described the mechanisms of STIX and TAXII, threat information exchange approaches and tools.

Zou *et al.* described a way how honeypots might get detected in botnets [33]. Also, Wang *et al.* described a similar approach [36]. This raised the question about the capabilities of honeypots. Should they be able to attack other systems and damage critical infrastructure to look authentic?

Bajtoš *et al.* described a method to monitor and analyse botnets wits honeypots [34].

Sokol *et al.* created a comprehensive overview about privacy issues by honey systems already including GDPR concerns [39].

Choo *et al.* described information culture and their usage in modern organisations [44].

Janczewski *et al.* described the principle of the “Need-to-know” approach, where information security in a company lies in disguising and hiding information; a security by obscurity approach [45].

4 Approach

To answer the key research questions, the intermediate questions must be answered first. Some assumptions on personnel and device assets, as well as on permissions and hideability of assets are made to investigate the intermediate questions and further the key research question. These assumptions are used to answer the intermediate research question from chapter 1, which will be confirmed via an experiment in the next chapter.

4.1 Personnel

The main question "How to hide a honey system to the inside?", as well as the question "From which staff can a honey system be hidden?" needs an overview of involved staff in a company or organisation. The following paragraphs enumerate a list of personnel in an organisation which will get in contact with a network device, resp. a honeypot and what the role know about the device. The following paragraphs list roles; this means multiple roles can be assigned to one person. As this list has been created with meticulous care, the list will probably not be complete to any organisation. The described rights and privileges are written from the point of view of an ideal organisation with an ideal assignment of permissions. When executing such a project, one must always consider the the state of the current assignment of permission and of the current culture of the organisation. The culture of an organisation rules the sharing of information inside and outside an organisation [44].

4.1.1 CEO / Secretary

This role is accountable for everything inside its organisation. If something miserably fails, the CEO or secretary is accountable for it as a last consequence [46].
This role needs to be aware of possible consequences in terms of a risk analysis to approve the project. The only reason to not involve this role, is when it is not trusted anymore. This implies a severe trust problem in the organisation and needs the project to be started by a supervisor. In generic deployments, the Chief

Executive Officer (CEO) or secretary needs to be informed at least, to comply with the business hierarchy. The CEO or secretary knows about the existence of the device and the possible risks, but does not need to know details, where and how it is implemented.

4.1.2 Lawyer

The lawyer advises about legal issues as described in section 2.5. The lawyer also will be needed if the honey system becomes rogue and starts to attack external or internal parties and indemnity claims or penalties need to be processed. The lawyer also gives advises on how much the auditor needs to know of the system. If the lawyer assumes liability of the honey system or the given advises depends on the contracts between the lawyer and the organisation, as well as on the applicable law.
The lawyer has the same knowledge level about the device as the CEO or secretary. The role of the lawyer knows about the existence of the device and the risks about it.

4.1.3 Accounting and Procurement

The accounting has an overview of the finances and assets of an organisation. The accounting sees all invoices of a company. The procurement purchases all assets. Both departments could ask for the purpose of the asset. If the purpose will ever get questioned by the department is a completely different topic.
The accounting and procurement only knows about the physical asset and/or software, if something has to be paid. In case of a virtual and free honey system, these two departments would not get in contact with it.

4.1.4 IT Administrator

The IT administrator is the most interesting role. The administrator is the role with the most knowledge about the IT infrastructure, which is often the most critical asset of an organisation [47]. Also, the role of the administrator has the most access to the systems and is granted with the most privileges and trust.
The administrator is supposed to know all IT related assets of an organisation. This role should have full insight in network traffic, as well full access to all IT systems, all physical and virtual machines. This access includes the access to general and specific documentation of company assets and processes. If something fishy or rogue appears inside the organisations network or some unknown is inside a server room, the administrator gets alarmed and triggers an investigation. With the rise of cloud computing, the significance of server rooms may shrink, but at least will stay present in producing organisations.
The administrator will see the asset, no matter if it is physical or virtual, as well as the traffic and electronically documentation of a network devices like a honey system.

4.1.5 DevOps

The DevOps will work on self assigned assets. The DevOps only knows IT assets from it's domain. When working with virtual machines, only required ones are visible and accessible. The DevOps gets partial access to the network to debug the developed and supported software. Also, the DevOps can have access to the general documentation.
The developer has no physical contact and no commercial contact to this device. The only way the developer notices the network device is, when it sends network traffic for whatever reason to a program of the developer.

4.1.6 Security

The security guard knows the building and and its people. This role notices any changes inside and outside a building. The security is in charge of the security cameras and needs to approve any modifications of the building.
The role of the security guard will notice any physical changes of server infrastructure. The security probable won't know what the device is for, although a small device on top of the server rack or hidden somewhere looks suspicious. If the honey system is virtualised, the security won't notice any change.

4.1.7 Product owner

The product owner or principal of the project knows anything about the project. Therefore, the honey system cannot be hidden from the principal. The principal should consult a lawyer whether the principal is entitled to lead this project and what are the responsibilities if anything goes wrong.

4.1.8 Random passerby

A random passerby could see the physical device or virtual device (if it is a privileged passerby) plus its cabling. The passerby could get curious but will not get suspicious about the device.

4.1.9 Machine Operator

This role operates a machine and knows how to debug basic failures. The machine operator knows the basic structure of the machine.
The machine operator notices a modification on the machine or if a new device will get directly connected to it and will ask questions if this will affect the performance of the machine and debugging process.

4.1.10 Data Protection Officer

The Data Protection Officer (DPO) knows all business processes around data processing, especially the ones with personally related data. The DPO lists and describes all processes and argues them to authorities requesting customers.

The DPO has access to all documentation of processes. This role is mostly entitled to question processes and ask details about them.

4.1.11 CIO

The Chief Information Officer (CIO) is accountable for the IT. The CIO knows about the documentation and has access to it. The CIO does not necessarily have contact with the device, as the role is accountable for the operating of the overall IT, but not a specific device.

4.1.12 CISO

The Chief Information Security Officer (CISO) is accountable for the information security of the organisation. This role needs to be informed and also has full access to the documentation. If this role is untrusted, the same principle like the CEO applies.

4.1.13 Auditor

The auditor could be an internal or external role. The auditor asks question about the organisations processes and assets. An internal auditor is used to self-reflect an organisation, an external auditor is used, when certifying the organisation or when the law obliges the organisation to do so.

The auditor knows the network and IT infrastructure. To determine, if a specific network devices needs to be mentioned during an audit and to what detail level, a lawyer should be contacted.

4.1.14 Personnel in the Audit

Personnel in the audit will possibly hear and read every information shared in the audit. The attendee should be aware of everything that will be discussed, otherwise a confidential audit could be a solution to avoid additional, unaware personnel.

4.1.15 Documentation Manager

The documentation manager has access to all documentation and requests relevant documentation. This role knows about a network device via its documentation.

4.1.16 Direct Supervisor

The direct supervisor is a role, that has to be created individually for every organisation. It will probably know most of the tasks of staff, but does not necessarily know every task and every asset. The accountability of the direct supervisor must be clarified for every organisation.

4.2 Properties of a Network Device

Each network device has some assets that will be enumerated. These assets will be visible to different roles inside a company. The following paragraphs explain the specific assets grouped into any presence or physical assets, network assets, and assets about the project or application.

4.2.1 Presence Assets

The first asset group is the simple presence of a device. The device can be *physically* present and visible through a classical pizza box, a Single Board Computer (SBC), cables in unusual colors or newly used switch ports, or just be a virtual device and therefore is visible through its *virtual* asset in a hypervisor.

4.2.2 Network Assets

The second asset group is about network and traffic. The *IP address* and the *MAC address* are fundamental assets of a network device. The Internet Protocol (IP) address represents the layer 3 address of the Open Systems Interconnection (OSI) model, while the Media Access Control (MAC) address represents the layer 2 address of the OSI model. They are essential when communicating over the network and need to be configured correctly. These assets appear in network traffic. The MAC address will appear in *arp watch* results and alerts, while the IP address appears in network configuration, like a firewall, an Access Control List (ACL), in connection tables, and Domain Name System (DNS) entries. The IP address often appears in inventory systems as well.

The second most important network asset is the *functional traffic* of the device. This is the traffic, that does the intended work of the service. In case of a web server, this would be the Hypertext Transfer Protocol

(HTTP). The functional traffic of a honey system, is the traffic that interacts with the attacker; the respective service varies with the honey system implementations.

Another important network asset is the *management traffic*. This traffic is used to manage the device, and the services hosted on the device. It is usually between an administrator or an administrative service and the service or OS of the device. Management traffic mostly consists of the services Secure Shell (SSH) and Remote Desktop Protocol (RDP).

Monitoring traffic is about to monitor the device and service states. Prominent services are Simple Network Management Protocol (SNMP) and emailing (Simple Mail Transfer Protocol (SMTP)). SNMP and the trap protocol are used to monitor the states, emailing is used to notify when something critical happens.

Synchronising traffic should not be forgotten. Synchronising traffic should be switched and routed over a separate network but often is not. Nevertheless, this traffic is used to synchronise the services on a network device for a High-Availability (HA) or Load-Balancing scenario. The protocols used include Common Address Redundancy Protocol (CARP), and the proprietary Virtual Router Redundancy Protocol (VRRP).

Logging traffic is about sending logs from a service or device to a log server or SIEM service. Most network devices, especially honey systems have a strong demand to send logs immediately to an external server (see subsection 2.4.1).

DNS traffic is used to resolve domain names to IP addresses. The device connects to a previously configured DNS server. The DNS traffic is seen as one of the most critical network services, as without a functional DNS service, most traffic requests cannot be resolved to the right address.

Another critical network service is the *Network Time Protocol (NTP) service.* It is used to synchronise time all over the internet. Without a correct time, most services won't work correctly and logging information may be useless.

Any *Domain traffic*, as Lightweight Directory Access Protocol (LDAP), the proprietary Active Directory (AD) and generic user authentication traffic may be configured on a device.

Connectivity checks, like Internet Control Message Protocol (ICMP) requests, can tell a lot about a network. These requests need to be taken in account, when hiding a device.

Traffic to *inventory services* appears, when configured. This traffic varies the used protocols, but is mostly an Application Programming Interface (API) over HTTP.

Apart from the traffic assets, the network device could be visible in the *inventory* software, in the *configuration settings* of DNS, the firewall, switch, and router.

4.2.3 Project assets

The last asset group, where the device appears is the *project* itself. The project also has three assets.

The *application* is the core asset, why this project has been started at all. This could include a web application, an API or similar.

The *technical documentation* is a tricky part, as the documentation could be shared across the organisation and cross-links to documentation may happen. The documentation of an asset is often not only in one central location, but distributed to multiple locations.

The *management summary* is presented to management staff to create a basis of decision making. It includes some basic information, a risk evaluation and possible affected processes.

4.3 Theoretical Information Obligations

Each organisation has a hierarchy which structures information flow and defines which roles need to know which information or is responsible and accountable for specific assets and processes. Table 4.1 visualise the theoretical information obligation, that allows a comparison of the matrix. It shows the access of all enumerated assets and roles of the previous sections and shows the access and knowledge level of the role about an asset of the honey system. The table is ranked from top to bottom from the most critical access to the least critical access to information. Critical information is ranked from left (least critical) to right (most critical). It is intended to give an overview about which role needs which access in a least privilege environment.

The following paragraphs answers the question “Which people need to be involved?” and describe the access to assets per role in detail.

4.3.1 Knowledge about true purpose

The *product owner* is the head of the project. The product owner knows everything about the project and therefore has access to everything. The product owner has access to the management summary and technical documentation describing the real purpose and therefore knows all things about the project. This role has access to the application itself (L7), as well as to any network related objects (L2 - L6), and knows the location of the physical or virtual device.

The *CISO* knows the real purpose of the project. The CISO has access to the management summary and as this role is accountable for information security, this role has access to the application.

	Presence		Network					Application		
	L1	Virtual	L2	L3	L4	L5	L6	L7	Documentation	Summary
Product owner	●	●	●	●	●	●	●	●	●	●
CISO								●		●
CEO										●
Lawyer										●
Supervisor										⊙
DPO										⊙
External Auditor										⊙
Personnel in Audit										⊙
Documentation Manager									⊙	
CIO									⊙	
Administrator	⊙	⊙	⊙	⊙	⊙					
DevOps	⊙		⊙	⊙						
Operator	⊙									
Security	⊙									
Procurement	⊙	⊙								
Accounting	⊙	⊙								
Passerby	⊙									
Internal Auditor										

Table 4.1: ● - Knows the real purpose of asset. ⊙ - Does not know the real purpose.

The table lists roles and assets derived from the OSI layers. Roles and assets are described in section 4.1 and section 4.2. Extended information about the matrix is given in section 4.3.

The *CEO* and the *lawyer* have access to the management summary including the knowledge of the real purpose. They need to know that there is such a project, but they don't need to now any technical details.
These roles have access to the most critical, sensitive information. As they have access to real information and documentation about the project, these roles must be reminded about the criticality of the information and the group of people who knows about the project.

4.3.2 Knowledge about fake purpose

The *supervisor* needs to know about a fake management summary. As the supervisor has access to human resources, this role needs to know what these resources are used for, and when they are available again.
The *DPO* needs to know that the project does something with security and it will deal with user data. It may be possible to justify this project by the means of perimeter security, so there is no need to identify a honeypot in any disclaimers and privacy statements. If so, it is sufficient to provide the DPO a fake management summary. Otherwise, the DPO needs to have access to the real management summary. This needs to be further defined by law. The DPO is always in conflict of privacy and security. To be fair enough, it is assumed, that an attacker won't complain about a honeypot at the data protection authority.
The *external auditor* as well as the *personnel in the audit* will probably only need a fake management summary. This depends on the topics of the audit and the answers given by a lawyer. If a fake management summary is not sufficient, the real management summary must be provided. The personnel knows, reads and hears anything that is discussed during the audit.
The *documentation manager* knows anything about the documentation and therefore will look for a documentation about the new system. The documentation manager therefore needs access to a fake technical documentation, just to satisfy the need of accessible documentation.
As head of the IT-department, the *CIO* needs a access to a fake documentation, to satisfy the requirements of a clean setup. It is not necessary to provide this role any kind of summary, as the CIO only needs to know that its systems are fully functional.
The *administrator* does not need anything to know about the project and application itself, but only needs knowledge up to layer four. This includes physical and virtual presence, MAC and IP addresses for routing purposes, as well as used ports for firewall access rules. Considerations must be made on the authenticity. A SSH honeypot in a server net seams possible, a modbus honeypot in a production net also seams possible. A RDP honeypot in a linux server net seams suspicious, as well as a generic TCP proxy honeypot (e.g. Glutton[1]) accepting connections to all ports. These indicators will not only make an administrator feel

[1] `https://github.com/mushorg/glutton`

suspicious, but also an attacker. The administrator does not know about the real purpose of the project.
The operator in *DevOps* has a limited set of knowledge about the project, only including assets which the role might get in contact with. This includes the physical presence of a server, the MAC address and the IP address. This role also has no knowledge about the real purpose of the project and does not necessarily need to know anything about the project.
If the network device is physically connected to a machine, the *operator* of the machine needs to get informed about the device. The operator only wants to know if this device interferes with its work and if the device must be considered while working or simple debugging.
The *security* personnel knows the physical appearance of the server room. Therefore it needs to know something about a physical device. If the device is standard server size, the security probably won't get suspicious, but the role will be when it is minicomputer sized. A new server in the room although will get the attention of the administrator.
The *procurement* and *accounting* know about the physical and virtual device if these have any financial costs. When using used hardware and free, open-source software these two roles do not interact with the project in any way. These roles also only need to know a very generic purpose.
A *random passerby* might notice a physical device. The role may be curious about it.

4.3.3 No knowledge

The *internal auditor*, in contrary of the external auditor, is the only role, that gets listed in this work, that has no need of knowing anything about the project. Still, to get a good internal audit report, it is advised to choose a person doing the internal audit, that already knows about the project through another role.
Any other role in the organisation does not need any knowledge about the project.

4.4 Hiding or disguise a network device

In order to hide a network device, one must be very carefully. It must be defined from whom this device should and must be hidden from and why. This exact definition is essential for the success of the whole project. According to the definitions, staff groups can be merged in order to hide and disguise the network device from the different roles. It is recommended that a network device is disguised, rather than completely hidden because of the simplicity of disguising compared to hiding it. Disguising and hiding a network device require two completely different approaches. This section will treat the questions "How to physically hide the system?", "How and where to log", and "How to hide traffic?".

When disguising a network device, a story and fake purpose must be defined for the device. For a honeypot, any common network device can serve as a fake purpose; a web-server for internal documents for example. A honeynet needs a better scaling device, which also creates a net: e.g. climate sensors. Climate sensors might be placed around the organisations building and measure all kinds of climate values; e.g. temperature, barometric pressure, humidity. It is possible to either place honeypots onto the existing climate devices and therefore enhance the function of them, or it is possible to enhance the network of climate devices with a variety of fake sensors. In terms of hiding the honeypot/honeynet, it is better to place them onto the existing devices, because no further devices are created. If this approach is also efficient to catch possible attackers it must be examined by every organisation, because of every organisations unique network topology.

When creating a story to disguise a network device, it is recommended to identify the staff groups which will most likely detect the real purpose of the project. Table 4.1 helps identifying them. The more about the application is known, the more likely it is that the real purpose is revealed. Also, the general knowledge about such devices in a company needs to be taken in account. Climate sensors may seem like a good story in a bank. Neither the management personnel, nor the administrators may know the functioning of a climate sensor. A climate sensor in a meteorological office may seem like a bad story, as staff is supposed to know the functioning of their sensors. A code name for the project is highly recommended. If the organisation already has a tradition of code naming projects, it is recommended to just apply to the already existing rules. A project with a code name has an official and confidential appearance. This is a good parameter for both person groups. Benevolent staff does not question the project, because of its official and confidential character, attackers and malicious staff are attracted by it. A disguised device still needs to look valid, therefore it is recommended to set any local settings (keyboard language, system language, login banner, required authentication method, etc.) equal or similar to the organisations standard.

When hiding a network device, other rules apply. A device does not need to be disguised, several staff groups must not even know that the device exists. Hiding a network device is a strict technical and psychological task. Whether a network device may be hidden from the management level is a pure legal question which will not be answered here. To technically hide it from the management level, it is mostly only necessary to simply not tell them or assign read permissions to the documentation. Hiding a network device therefore is limited to its physical and virtual appearance. To hide a virtual device, the hypervisor must be assigned with correct permissions. This might be the case in large organisations, but Small and Medium Businesses (SMB) won't limit administrators in working on the hypervisor. Hiding a physical device requires knowledge of the physical infrastructure of the building. Hiding a smaller device might be easier than a bigger device, therefore a SBC is recommended. A SBC can be tapped to the inside of a productive machine, or in a

distributing cabinet. It is also recommended to place such a device in a server rack from another department. This triggers a psychological effect: “I am not in charge”. The own department does not question the device, as it is in the authority of the other department. The other department is afraid to access the device, because it is not from their department. This setup requires two separate departments. It is also of advantage, if there is a rivalry between these departments and they don’t communicate good. When placing such a device, it is necessary to adapt it to the environment. For example, when it gets connected to a switch, where only blue cables are wired to, connecting it with an orange cable seems suspicious.

Hiding a device at the network layers is a tough challenge. When traffic is not monitored via a SIEM or similar, this point may be neglected. To hide the network traffic from a network device, it is necessary to reduce its traffic. To reduce traffic, a host firewall (e.g. iptables) can be used to block or drop all outgoing traffic, except strictly necessary one. The only necessary traffic of a honeypot is the functional honeypot traffic, as well as time synchronisation for logging purposes, logging, and monitoring. Logging and monitoring of a honeypot must be done in real time, as honeypots bear a great danger of compromisation, while the management of a honeypot could be done via physical access to it. Functional traffic is only allowed to be initialised by a remote host, while logging is only allowed to be initialised by the honeypot. Monitoring depends on the implementation and needs to be considered by the organisation. An approach to effectively hide traffic would be to use two network interfaces. One for the honeypot traffic, and one for management traffic. The functional Network interface controller (NIC) could be connected to a functional network while the management NIC is connected to a dedicated management honeynet. If the honeypots in this honeynet are placed in different networks, this concept may compromise the organisations overall security. A compromised honeypot may be used as a jump host over the management network to other networks, bypassing perimeter firewalls! Otherwise, to avoid logging of management traffic, the usage of available resources inside the same network is recommended: NTP-Server may be in the same network, as well as a distributed logging server may push logs from network A to the central logging server in network B.

Nevertheless, the first impression of a network device counts. Configure the device, as similar, as any other network device in the company, ergo set a correct Message of the Day (MotD), a correct SSH banner, correct hostname and correct timestamp.

4.4.1 Hide from security services

Organisations may have systems that regulary scan and access network resources, which might not be desired on specific devices. In order to hide the network device, resp. the honeypot, it should not respond to any requests coming from these systems. A host firewall (e.g.: *iptables*) is suitable to just drop packets

from specific hosts without the need of adjusting any other system. It is recommended to drop packets from at least the following systems, to hide the network device: automated vulnerability network scanner (e.g.: OpenVAS, Nessus) and inventory systems. Monitoring systems need to be considered carefully. It may be necessary to restrict access from these systems to the honeypot, or it may be desired to allow access to disguise the honeypot as a productive system. The following enumeration describes the most common services and how to consider them when hiding a network device.

- Vulnerability scanner: A vulnerability scanner like Nessus or OpenVAS automatically or manually scans the network for vulnerabilities. When a honeypot, offering vulnerable services is being scanned, the vulnerability scanner would alert several staff roles, including the administrator. It is recommended to blacklist owned vulnerability scanners, therefore drop requests via a host firewall.
- 802.1x Authentication: Networks may restrict access based on 802.1x rules watched by a radius server. These rules may base on MAC addresses, usernames and passwords, and cryptographic certificates. To successfully connect to such a network, it is necessary to announce the device to the radius server. The MAC address based concept could be bypassed by assigning the honeypot an address, already existing in the network. This is highly unrecommended, because it bears the risk of a MAC-address collision. When doing so, select a MAC-address of a device, which has already been decommissioned and will not join the network anymore. A username/password concept would not leave any traces on the radius server itself, but at the LDAP/AD server. The certificate based authentication requires a certificate, most likely issued by the local Public Key Infrastructure (PKI). This also, does not leave any traces on the radius server but at the local Certificate Authority (CA).
- Arp-watch: An arp-watch monitors the network about device appearances and alerts administrators if a new device with a new MAC-address has been found. The same rules apply, as with the radius server and MAC-addresses. The radius server and arp-watch may be the same product, but they may as well be splitted. When submitting the MAC address of the new device to the list of known devices at the arp-watch, the arp-watch would not alert anybody when seeing the device. An administrator may overlook such an entry very easily.

To successfully hide network traffic from a device, there are two approaches:

- Not sending any traffic
- Hiding functional traffic in noise

Not sending any traffic seems difficult, as network devices are built to send traffic. An approach may be to not send anything on the functional device but only listen. In Operational Technology (OT) networks, it may be possible to exchange a switch with a hub to monitor all traffic passing the hub. This impacts performance

a lot, as most “modern” hubs have a maximum throughput of 100Mb/s per Hub which decreases with every communicating member on the hub. The hub may be exchanged during a Business-Continuity-Management (BCM) test and be just forgotten by purpose. Hiding functional traffic under a lot of noise is a good way to tire an administrator. If a lot of noise is generated by the device, it may happen, that the administrator marks all traffic as noise and won't notice the functional traffic.

4.4.2 Hide Documentation

To hide the documentation and process assets is a critical part of hiding a network device. The documentation of a network device includes everything known to the organisation to the purpose and implementation of the system. This section help answering the question “How and where to document everything about the device?”.

When disguising a device, a real and a fake documentation will be created. The real documentation contains everything a normal documentation contains as well. The real documentation needs to be hidden either via permission sets or via storing them into another storage, out of reach for generic organisation staff. The fake documentation needs to be placed according to the organisation policy. Therefore when doing a commissioning process, the inventory, as well as any alerting, entries in host lists, etc. remains the same to the organisations process, solely the real purpose of the device gets disguised. Any shared access credentials may be created and submitted to the organisations credential collection (password safe, etc.) to reveal any curious staff. The code of the project should not be submitted to any automation software, but submitted to a specific Version Control System (VCS) project with a special permissions set. Updating the fake documentation has an advantage and a disadvantage. When updating the documentation, the device looks more like a real up-to-date device, but also reminds administrators of the device.

When hiding a device the procedure for the real documentation is the same as when disguising a device. Hide the documentation via a special permission set or store it into another storage out of reach of generic staff. As the device gets hidden, there will be no fake documentation, therefore no public available information about it. Changes in the infrastructure required for the project that require a documentation need to be avoided or documented with special permissions, or if not possible, not documented which is not recommended. This will raise issues and therefore the concept of hiding a network device needs to be evaluated if it is possible at all to do this in an organisation. Not documenting infrastructure settings like IP-addresses, switch ports, firewall configuration, etc. will lead to inconsistencies (double-assigned IP-addresses, overwritten configuration, etc.) and availability issues. Keeping an inventory up to date public available is a critical

control in IT security[2].

4.5 Goals

When hiding an asset from personnel, it must be clearly defined why doing so. What are the goals when hiding an asset. What type of attacker is the asset hidden from:

- External attacker
- External attacker, already operating from the inner network
- Internal attacker

4.5.1 External attacker

An external attacker is the easiest one to identify and detect from the above list. An external attacker does only know very limited information about the target and needs to gain these information in the first place. To catch an external attacker it is only necessary to hide a honeypot to external people. Therefore, all technical actions must be made, but there are no organisational actions necessary.

4.5.2 External attacker, already operating from the inner network

An external attacker, already operating from the inner network has successfully penetrated the security perimeters and is able to jump to hosts in the inner network of an organisation. This type of attacker may have fully access to technical parts of an organisation but has only limited access to organisational knowledge. This attacker has technical access to the organisation, but the only knowledge is gained from documentation placed into the organisations IT-systems.
To hide a honeypot from this type of attacker it is necessary to hide the documentation of the honeypot according to subsection 4.4.2. Also, the usage of code names for the project is recommended. The usage of a need to know principle [45] may prevent unintended, uncontrolled information spread.

4.5.3 Internal attacker

To detect an internal attacker is the most difficult thing. An internal attacker could be staff, or a someone getting information from an insider. An insider is the most dangerous attacker, as the insider is capable of doing the most damage in little time. Trying to catch an insider requires a lot of privilege management which may create a bad mood in the organisation. Trying to catch a possible insider means to mistrust staff.

[2] https://www.cisecurity.org/controls/inventory-and-control-of-hardware-assets/

Actions to catch insiders should be strictly limited temporarily and only be done when absolutely necessary. They include all technical specifications, as well as organisational actions.

4.6 Possible project assignee groups

When planing to hide or disguise a honey system it is necessary to also plan the project assignee. Who will implement and maintain the project. This paragraph helps deciding it. What are the possible constellations of project assignees? Which constellations bear which advantages and which risks?
There are three possible and useful group constellations; single, one team, and two teams; which will be described in the following. All three approaches only work if the organisation is used to confidential projects. Everything else looks suspicious.

4.6.1 Single person

A single person gets instructed to implement and hide a honey system. This person need to be trustworthy, an expert on honey systems and an expert of the organisations IT infrastructure. This person cannot get any internal help and needs to decide any technical questions autonomous. This approach has the advantage of low costs, but the disadvantage of a low liability for the assignee as well as for the organisation. The organisation has no way to monitor the doing of the assignee, as well as the assignee cannot enter any objection to an accusation of the company. Also, if the approach is busted, the trust of other colleagues to the administrator will be ruined and the performance of collaboration will decrease. The single person approach is the cheapest one, but not the best one. It is only recommended in small organisations, not having multiple IT teams.

4.6.2 One team

Only one team gets to operate a honey system. This means that everybody, except the one team and personnel from Table 4.1 does not know anything about the project. This requires either already existing multiple IT-teams, or the creation of a secret team in a team. The advantages are a better liability compared to the single person approach, for both the organisation and the single person. It is not possible to accuse a single person about anything secret, that only few persons can check, also, the organisation has a better tool to control the single person about its doing. Also the complexity may decrease, depending on the previous existence of the team. If the team already existed, the project becomes less complex, as roles inside the team are already defined and the IT infrastructure already has non-suspicious privilege and permission presets.

If the team is only created for the project complexity rises compared to the single person approach, as all these roles, privileges and permissions need to be newly defined. The disadvantages are a worse liability of the company compared to the two teams approach and a newly created feeling of privileged administrators. Personnel not inside the team are automatically labeled as not trustworthy. If anything goes wrong and this approach gets discovered this creates a huge disharmony inside the IT department which will have a massive negative impact on performance. The IT department will not work together anymore and a lot of personnel need to be exchanged. This again will lead to a massive negative impact on performance. Therefore this approach is not recommended, as it bears to much risk.

4.6.3 Multiple teams

The multiple team approach is the most expensive, but also most controlling approach. Multiple teams, at least two, operate their own honey system. The other teams know that another team is operating a honey system but they do not know where and who. This enables to control every team in the organisation by somebody else. This only works if at least two IT-operating teams work in an organisation and they have a clear domain of responsibility. It also needs already existing systems that one team has no access to, so honey systems can be hidden. This approach denies the massive disadvantage of distrusting other personnel, as the approach will become a security improvement in company culture. The disadvantages are the higher costs and that at least two teams are needed which are not available everywhere. This is the recommended approach.

4.7 Classifing Honey Systems about their Hiding Potential

The previous paragraphs help to rate honey systems about the hiding potential which will be discussed in the following:

1. *Honeytokens* are the easiest honey system to hide. As they are just plain data or files, they only possess two assets according to section 4.2, the presence and project asset, but no network asset. Therefore there is no need to hide any network asset, as well as there is no need to document any associated asset (which needs to be hidden as well). The honey token therefore only possesses a physical or virtual asset, as well as a management documentation and technical documentation which need to be hidden. The hiding and disguising potential of honeytokens is equally set.
2. *Honeypots* rank second as they are a single device or service which need to be hidden. Nevertheless, Honeypots possess the full range of section 4.2 assets which need to be hidden or disguised.

Honeypots are easier disguised than hidden, because of their network assets.

3. *Honeynets* are hard to hide. Because of their amount of honeypots, this requires a lot of organisational thoughts and is best suitable for larger companies with a distributed set of permissions. Internally hidden honeynets might only be used in a multiple team approach (subsection 4.6.3). Honeynets are easier to disguise than honeypots (e.g.: a distributed climate sensor control network in a production facility seems more valid than a simple sensor), but are harder to hide than honeypots because of their increased number of devices.
4. *Honeywalls* are the hardest system to hide. On the one side, they only make sense when operating a honeynet which is already hard to hide. On the other side, honeywalls operate as firewalls and route traffic. It may be impossible to hide a honeywall, but only be possible to disguise it, as also valid traffic need to know the existence of a honeywall.

As seen, the type of assets as well as the number of devices and purpose of the system play a significant role in rating honey systems about their hiding potential. An interesting part is, that hiding a system gets harder with an increasing number of devices, while disguises gets easier with an increasing number of devices.

4.8 Busted?

What happens when the purpose of the project is busted? Someone revealed the objective of the network device. This section answers the question “What to do when the device is being found by an employee?”.

When someone busted the honeypot, the assignee or product owner needs to act fast. The respective person needs to be privately spoken to immediately. It is necessary to explain the whole project in detail to the person, to not create any rumours and to not create the feeling to fob the person off. This includes the purpose, the aim and the assigned persons. Also, it must be explained that this is confidential information. If a Non-Disclosure-Aggreement (NDA) is necessary needs to be defined by every organisation itself.

What happens if too many people discover the true purpose needs to be defined by every organisation, as well as the definition of “too many people”. It is recommended to define the threshold per department and by the company. The decision, if the project should be revealed, terminated, or continued also depends on the goal of the project (section 4.5). To catch external attackers operating from the inside, it is recommended to either reveal or continue the project, while when catching insiders it is recommended to terminate the project, as the goal has not been achieved. Retrying such a project should only be done after a cool down phase, when people already forgot the project or have fluctuated much in the organisation.

4.9 Busted!

The goal of the project will lead to someone getting busted. But what happens, if the honeypot raises an alert?

First of all, the alert needs to be verified. Is this a false positive or a real incident. Then, it needs to be distinguished if the incident is a real attacker or just some curious in the organisation. If the incident seems to be someone curious, monitor the behaviour and decide if actions according to section 4.8 are necessary. If the incidents originates from an external attacker, actions according to the organisations policy or other best practices should be done without mentioning the honeypot when warning other, not inaugurated personnel.

To distinguish between curious personnel and an insider threat, it is recommended to ask the person specific direct questions about the happened access attempt. Why did the person try to access the system? How did it find it? What did the person expect to find? After this interview either the actions of section 4.8 are necessary, or to treat the person as an insider threat according to organisation policy.

5 Experiment

An experiment was made to see how to hide a honeypot in an organisation. The research questions of this experiment is related to the research questions of this thesis. Does someone in the organisation discover the real purpose of the device? If so, how long does it take to realise the real purpose? Whom will it be? What is the initial trigger? What is the workflow when finding the device.
The hypothesis is that the staff will comply to the theoretical information obligations of section 4.3 and therefore only chosen staff knows the real purpose of the device as a honeypot.

5.1 Setup

A physical device, a Raspberry Pi v3 was chosen to serve as host of the honeypot. Luckily, one device was already present, so no new device had to be purchased. Therefore, there was no need to inform procurement and accounting. The Raspberry Pi was placed into a floor network distribution cabinet. Therefore it was not placed inside the server room, where a lot of changes happen, but placed into a very static place which is still controlled by the IT personnel. As seen in Figure 5.1 it was placed in the back of the box, not clearly visible to everybody and labeled with the hostname. The hostname was chosen to meet the requirements of a code name of a project of the organisation. As used code names are names of famous mountains, a non-used mountain to name the project was chosen: Project Mount Fuji, for example. This should attract malicious user and distract compliant members of the network. Naming a project with a code name, enables it to document common assets easily, like IP addresses (192.168.0.250 is the address of asset fuji01.example.org). The Raspberry Pi was setup with a standard Debian image. As a honeypot, cowrie[1] was chosen. Cowrie is a SSH and telnet honeypot written in python. The chosen implementation was the docker version of cowrie, so docker was installed on the Raspberry Pi. To successfully run docker on an older raspberry, some bash commands must be executed to run iptables in legacy mode, which are show in Listing 5.1. The settings to configure arptables and ebtables may not be needed, depending, whether they are installed or not.

[1] `https://www.cowrie.org/`

Figure 5.1: The Raspberry Pi from the experiment placed in the back of a network distribution cabinet. On the left side, some network devices are stacked. It is placed behind a cover, so it is not visible when looking through the glas door or even when opening it.

Listing 5.1: Setting iptables back to legacy version

```
update-alternatives --set iptables /usr/sbin/iptables-legacy
update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
update-alternatives --set arptables /usr/sbin/arptables-legacy
update-alternatives --set ebtables /usr/sbin/ebtables-legacy
```

The raspberry was inserted into the companies automation system, so the "essential" services and settings (MotD, NTP, DNS, hostname, syslog, monitoring, etc.) are configured similar to already existing systems in the organisation. Also, the password for fallback users were stored via the standard organisation processes. Raspberry Pi computers use ARM architecture, the docker images of cowrie were only compatible for the x86 architecture. Therefore the Dockerfile of cowrie[2] was slightly adjusted, as seen in Listing 5.2. The architecture was set to *arm32v7* and *cargo* was added to be installed in the builder image. Cargo[3] is the package manager for Rust and is required to compile a *wheel*[4] file for the *cryptography* packet of python.

[2] https://github.com/cowrie/docker-cowrie/blob/master/Dockerfile
[3] https://doc.rust-lang.org/cargo/
[4] https://pythonwheels.com/

Listing 5.2: Slightly adjusted Dockerfile to install cowrie

```
1  # This Dockerfile contains two images, 'builder' and 'runtime'.
2  # 'builder' contains all necessary code to build
3  # 'runtime' is stripped down.
4  ARG ARCH=arm32v7
5
6  FROM ${ARCH}/debian:buster-slim as builder
7  LABEL maintainer="Michel Oosterhof <michel@oosterhof.net>"
8
9  WORKDIR /
10
11 ENV COWRIE_GROUP=cowrie \
12      COWRIE_USER=cowrie \
13      COWRIE_HOME=/cowrie
14
15 # Set locale to UTF-8, otherwise upstream libraries
16 # have bytes/string conversion issues
17 ENV LC_ALL=en_US.UTF-8 \
18      LANG=en_US.UTF-8 \
19      LANGUAGE=en_US.UTF-8
20
21 RUN groupadd -r -g 1000 ${COWRIE_GROUP} && \
22      useradd -r -u 1000 -d ${COWRIE_HOME} -m -g ${COWRIE_GROUP} ${COWRIE_USER}
23
24 # Set up Debian prereqs
25 RUN export DEBIAN_FRONTEND=noninteractive; \
26      apt-get update && \
27      apt-get install -y \
28          -o APT::Install-Suggests=false \
29          -o APT::Install-Recommends=false \
30        python3-pip \
31        libssl-dev \
```

```
32          ca-certificates \
33          libffi-dev \
34          python3-dev \
35          python3-venv \
36          python3 \
37          python3-cryptography \
38          gcc \
39          git \
40          build-essential \
41          python3-virtualenv \
42          python3-wheel \
43          cargo \
44          libsnappy-dev \
45          default-libmysqlclient-dev && \
46      rm -rf /var/lib/apt/lists/*
47
48  # Build a cowrie environment from github master HEAD.
49
50  USER ${COWRIE_USER}
51
52  RUN git clone --separate-git-dir=/tmp/cowrie.git \
53      https://github.com/cowrie/cowrie ${COWRIE_HOME}/cowrie-git && \
54      cd ${COWRIE_HOME} && \
55          python3 -m venv cowrie-env && \
56          . cowrie-env/bin/activate && \
57          pip install --no-cache-dir --upgrade pip && \
58          pip install --no-cache-dir --upgrade wheel&& \
59          pip install --no-cache-dir --upgrade cffi && \
60          pip install --no-cache-dir --upgrade setuptools && \
61          pip install -r ${COWRIE_HOME}/cowrie-git/requirements.txt && \
62          pip install -r ${COWRIE_HOME}/cowrie-git/requirements-output.txt
63
```

```
64 FROM ${ARCH}/debian:buster-slim AS runtime
65 LABEL maintainer="Michel Oosterhof <michel@oosterhof.net>"
66
67 ENV COWRIE_GROUP=cowrie \
68     COWRIE_USER=cowrie \
69     COWRIE_HOME=/cowrie
70
71 RUN groupadd -r -g 1000 ${COWRIE_GROUP} && \
72     useradd -r -u 1000 -d ${COWRIE_HOME} -m -g ${COWRIE_GROUP} ${COWRIE_USER}
73
74 RUN export DEBIAN_FRONTEND=noninteractive; \
75     apt-get update && \
76     apt-get install -y \
77         -o APT::Install-Suggests=false \
78         -o APT::Install-Recommends=false \
79       libssl1.1 \
80       ca-certificates \
81       libffi6 \
82       procps \
83       python3 \
84       python3-distutils && \
85     rm -rf /var/lib/apt/lists/* && \
86     ln -s /usr/bin/python3 /usr/local/bin/python && \
87     ln -s /usr/bin/python3 /usr/local/bin/python3
88
89 COPY --from=builder ${COWRIE_HOME} ${COWRIE_HOME}
90 RUN chown -R ${COWRIE_USER}:${COWRIE_GROUP} ${COWRIE_HOME}
91
92 ENV PATH=${COWRIE_HOME}/cowrie-git/bin:${PATH}
93 ENV COWRIE_STDOUT=yes
94
95 USER ${COWRIE_USER}
```

```
96  WORKDIR ${COWRIE_HOME}/cowrie-git
97
98  # preserve .dist file when etc/ volume is mounted
99  RUN cp ${COWRIE_HOME}/cowrie-git/etc/cowrie.cfg.dist ${COWRIE_HOME}/cowrie-git
100 VOLUME [ "/cowrie/cowrie-git/var", "/cowrie/cowrie-git/etc" ]
101 RUN mv ${COWRIE_HOME}/cowrie-git/cowrie.cfg.dist ${COWRIE_HOME}/cowrie-git/etc
102
103 ENTRYPOINT [ "cowrie" ]
104 CMD [ "start", "-n" ]
105 EXPOSE 2222 2223
```

This docker image is the base image which is adapted with various configurations. The configuration files can be seen at the readme file of the project[5]. These files need to be copied to the allocated places, which is why further modified docker images of cowrie were made. Listing 5.4 shows an example which only uses the standard configuration file. The image *mount-fuji* represents the standard image build before with the command seen in Listing 5.3. When modifying the standard image via another dockerfile, it enables the operator to adjust the honeypot without continuous access to the internet. The standard image is fetched locally; only when updating the software, external resources are required.

Listing 5.3: Build a docker image

```
1 # -t specifies the tag of the image
2 # . represents the location of the Dockerfile
3 docker build -t mount-fuji .
```

Listing 5.4: Modify docker image of cowrie to configuration

```
1 FROM mount-fuji
2 COPY cowrie.cfg /cowrie/cowrie-git/etc/
3 COPY --chown=cowrie:cowrie entry.sh /cowrie/cowrie-git/
4 RUN chmod a+x /cowrie/cowrie-git/entry.sh
```

The honeypot was set to listen to the standard SSH port 22/tcp, without telnet. The management ssh port was a custom port. Which port to use as a lure port depends on the organisation. The lure port (the port on which the honeypot listens) should be the standard SSH port. If the company uses 22 as a standard, this

[5] https://github.com/cowrie/cowrie

port should be used as lure port and a custom port should serve as management port. If the company uses a custom port as a standard, the custom port should be used as lure port and another custom port needs to be the management port. Never use the port 22/tcp as a management port for the honeypot, as attackers would find this port very quickly. As part of the hardening, the management access was restricted to workstations of the administrators, the lure port was set open to the internal network.

To ensure, that the honeypot always runs, even after a reboot, a simple docker-compose file (Listing 5.5) was called. It defines a service *fuji01*, which depends on the latest image *mount-fuji* and binds the host port 22 to container port 2222.

Listing 5.5: Docker-compose.yml to keep the container running

```
1 version: "3.3"
2 services:
3   fuji01:
4     image: "mount-fuji:latest"
5     entrypoint: ./entry.sh
6     ports:
7       - "22:2222"
8     restart: "always"
```

To log the usage of the honeypot, the docker daemon was configured to send anything written to stdout in a container to the syslog daemon of the host. Cowrie prints any log to stdout. No logging configuration of cowrie itself was used. The modified entry point directs to a script (Listing 5.6) that aims to exclude all sensitive strings given from logs. As the honeypot prints all user entered strings to stdout which get send to log servers, sensitive information must be prevented on being printed to stdout. This could include common passwords, as administrators are not aware of the honeypot and may try to login with these. Only write part of the sensitive string, to not disclose the whole string inside a common attacked asset. The command *grep -vi* removes any line with any case insensitive string matching.

Listing 5.6: Modified entry script (entry.sh) to exclude sensitive strings from logs

```
1 #!/bin/bash
2 cowrie start -n | grep -vi <Part of a sensitive string>
```

To monitor the usage of the honeypot, an alert was implemented into a monitoring system. Any occurrence of the strings “cowrie” in combination with “new connection” triggers a real time alert being send to the respective administrators for further investigations.

5.2 First contact

While the setup of the honeypot was made, the first contact of the honeypot with the organisation happened. While some personnel was present in the office, some were working from home and some were on holiday. Next a timeline of what happened on the first day is shown.

- Day 1, 09:00 - Asked about some Unix questions, administrator A (who is in the office) asks what this is all about. When answered this would be a project for manager B, no further questions were asked.
- 10:00 - When issued a ticket about a firewall change and assigned to manager H (who is not in the office), administrator A, who gets all firewall tickets as an email copy mentions, that he finds no commission checklist about the requested server. The checklist already exists, but administrator has no access to it. Administrator A is answered with "Oh, well, yes. I am going to do it now".
- 14:00 - Administrator A mentions, the checklist still lacks.
- 14:10 - A ticket about the server commissioning is created, which is mandatory due to the checklist. This was not intended in the first place, but made due to the requests of administrator A.
  This triggers a massive reaction. Administrator A gets suspicious and starts an investigation. As the assets is documented in the IP list and an entry exists at the DNS server, he starts trying to connect via SSH, RDP and looks for the server in the hypervisor overview. (SSH honeypot is not productive at the moment.)
- 14:20 - Administrator A starts looking for the server in the server room. He mentions, that this might be a Raspberry Pi, as he saw some new ARM configuration in the automation framework.
- 14:30 - Administrator A searches the SIEM. As the honeypot is not productive, no specific logs, other than generic network traffic (DNS, NTP, etc.) is seen.
- 14:41 - Administrator M (who is not in the office) requests a better description of the use case of the server in the ticket issued at 14:10.
- 14:42 - A request for permission to access the exact checklist in the documentation system is issued by administrator A. This means, the site has been found. He admits, that simple URL searching made him find it, as all checklists have the same prefix and the hostname as URL. Also, the documentation system did not return a 404 not found error message, but a 401 authorisation required.
- 14:47 - Administrator A starts searching all switches for newly connected devices, but cancels, as it takes too long, and he does not know what to look for exactly.
- 15:00 - Administrator A finds the open management port, but fails to login.
- 15:15 - Administrator A is told to search the management password safe again for credentials.

- 15:30 - Administrator A mistakes the honeypot for a *conpot*[6] honeypot, as the bash history lists a conpot command.
- Day 2, 10:00 - Administrator A coincidentally finds the correct MAC-address in a firewall log but does not investigate it any further.

The first contact to the device was due to an erroneous ticket announcement by the author of this study. This ticket was not intended to be sent as an email to a group of administrators, but only to manager H, who is the product owner. Therefore the first contact was already early. This triggered the request for documentation. The provided documentation was only a project name (Mount Fuji), which did not satisfy the present administrator. There was to few information and to much shady pieces of information placed which triggered the investigation. A fake description could have prevented this. The investigation itself was executed very fast, which left no time to adjust anything, but only to log the happening. At the end of day 1, administrator A already knew the real purpose of the device, but does not know the location of it. As a result of this investigation, it is recommended to setup the device completely offline or of company and implement it into the companies network already fully operational.

5.3 Second contact

When the setup was done, a second contact from another team happened with the system. Again, a timetable is presented. This timetable also results from log entries.

- Day 7, 10:50-11:10 - A login attempt on the honeypot happened by administrator C.
- 11:40 Manager F asks manager H about the asset. Manager F is the supervisor of administrator C. Manager F and administrator C already know its a honeypot. Manager F mentions, that the project description lacks detail and administrator C was wondering about the backup concept and where it is located.
- 11:40-11:50 - A successful login attempt. Administrator C starts searching the file system.

After examining the log entries, administrator C was asked about the events. When coming back from a holiday, he was triggered by the undetailed ticket about a new server commissioning. The server description in the ticket was only the project name, which did not satisfy him. He did not receive any email but browsed through older tickets to catch up what happened during his absence. Although, it was clearly a server with a code name and restricted documentation, he thought that being part of the infrastructure team, he needs to know the purposes of any servers present. Administrator C asked his manager F, if he knows

[6] `https://github.com/mushorg/conpot`

something about it. Manager F questioned manager H, the product owner, about the asset, but manager H only returned a generic answer. Manager F was satisfied with the answer and returned it to administrator C, who was not satisfied and started a further examination. He asked administrator A, an administrator of another team, about the asset, who answered also with a generic response, which still did not satisfy C. The following consideration was made. The asset is in the standard domain and accessible over SSH, therefore the standard backup SSH password should grant access. Administrator C logged into the system with this password on port 22, which is the honeypot. The honeypot is configured to grant access with any user/password combination. After seeing some old system descriptions including */etc/os-release*, indicating a debian 7 (Wheezy), the first assumptions on a security system was made. When logging out and in again with generic credentials, it was clear that this is a honeypot.

Again, the undetailed description in the ticket, resp. the ticket itself, triggered the initial investigation. Although, administrator C did not receive an email about it, he still found it in the ticketing system while browsing through old tickets. The first questions about the asset were raised to the direct manager, who was satisfied with a generic answer, but this answer did not satisfy an administrator. As the honeypot was poorly configured, it was busted very fast and the real purpose of the asset was clear to the administrator.

5.4 Third contact

The third contact differs very much in the behaviour of the administrator. Administrator M (14:41 from the first contact) asked the project assignee about the project name and the purpose, three weeks after the initial contact during a lunch break. He was told to ask again the next day, when only the selected participants are present. Administrator M noticed the project via the erroneous ticket. After a quick research, he found some bread crumbs in the automation system, but did not guess the purpose of the project. He admits to already asked administrator B about the project who also did not know anything about it and tried to find a web Graphical User Interface (GUI) in a browser. Due to lack of time and interests, administrator M stopped investigating at this point. When being to that this is a confidential project, administrator M lost completely interest and did not investigate any further.

Once again, the initial trigger was the erroneous ticket announcement. Due to the lack of access to other information and the lack of time resources, it was not investigated any further. The purpose of the network device is still unknown to administrator M.

5.5 Qualitative evaluation

The described contacts match in some details and differs in others. This will be evaluated in the following paragraphs. Unfortunately, only the first contact is documented with detailed timestamps to clearly understand the approach of investigation. No investigation was carried out simultaneously to another one. Participants who already knew the real purpose did not declassify it to requesting other parties.

- The ticket: The initial trigger was the ticket at every contacts. The ticket started all investigations. The difference is, that an email announcement started the investigation twice, while the sole existence of the ticket started the investigation once.

  The ticket was part of the standard procedure for installing a new server and was chosen to do so, to make the honeypot look more valid and as part of the documentation of the server. To mitigate such a trigger, the documentation could be separated completely, or complex permission settings could be applied to the ticketing system. Also, announcing such tickets via email or similar must be avoided.
- Investigating staff groups: The only investigating staff were administrators. No other group of Table 4.1 had noticed anything unusual and asked questions or investigated on their own.
- Location of the investigating staff: All three investigator were at a different place at the time of the start of the experiment and therefore while the ticket was issued. Administrator A was in the office, next to the executive of the experiment. Administrator M was in home-office at that time, while administrator C was on vacation and therefore started the investigation later on. The hypothesis, that older information (tickets, emails, etc.) are not interesting when coming back from a holiday has not been confirmed.
- Starting time of the investigation: All three investigations differ in the starting time. Administrator A started immediately after the ticket issuing. Administrator M noticed the device, when the ticket was issued, but did start a first questioning weeks after it. Administrator C started the investigation a week after the ticket issuing, when returning from a vacation.
- Contact person: Administrator A asked the project assignee directly, administrator M asked administrator B, and administrator C asked manager F. The first contact persons all have in common to be in the same room, when the device is detected. Then, the chain of contact differs. Administrator M asked the project assignee after an unsatisfactoring answer from administrator B, while manager F asked manager H about the project and returned the answer to administrator C. Staff prefers to talk to people at the same level (the project assignee is also an administrator).
- Order of investigation: Administrator A and C started searching the documentation, connecting via

common management methods (SSH, RDP, etc.) and finally asked the contact person. Administrator M checked the documentation first, but asked the contact person without connecting first. Questioning the assignee was always the last option. Manager F although directly asked manager H. No consultation of the documentation was made.

- Connecting: To connect to the device, SSH on various common ports and RDP has been tried. Also, it has been looked for the device on the hypervisor to directly connect to.
- Physical access: Administrator A and C did not find any occurrences of the device in the hypervisor and assumed it to be a physical device. Therefore, administrator A started an investigation to search the server room.
- Duration of investigation: The duration of the investigation of administrator A was about an hour. The investigation of administrator C lasted for half a day, while the investigation of administrator M lasted around an hour over three weeks.
- Intention and intensity of investigation: Administrator A was simply interested in the challenge and compared it to a Capture-the-flag (CTF) contest. His intensity of investigation was rather high. Administrator C wanted to secure the full knowledge of his understanding of the network and server infrastructure. His investigation was also quite intense. Both revealed the true purpose of the device. Administrators M intention was to ensure, that it is no malicious device. His investigation was not intense and he did not reveal the true purpose. Manager F did not do an intense investigation, as he was only asking manager H about administrator Cs request. Manager F also did not reveal the true purpose. The more a personal nature an investigation is, and the more curious a person is, the more likely it is to reveal the true purpose.

To summarise the enumeration, the announcement of the ticket triggered some administrators but not all, otherwise the sole existence of the ticket triggered also an administrator. The approach to submit such tickets when sound administrator are out of office failed, as the ticket was found when returning from vacations. An interesting result is, that always the person next in the office has been the first contact, although the corresponding contact for that project has always been clear. Also, the different staff groups preferred to question their own staff group. The administrators asked an administrator first, while the managers asked a manager first. The administrators also questioned the documentation first, tried various things to connect to the device and asked somebody last, while the managers asked somebody first and then questioned the documentation.

When considering all various detection vectors, the documentation and routine was the initial detection trigger. To hide or disguise the documentation is an essential and critical part of such a project. Most assets

of section 4.2 have been discovered and revealed, all being either organisational or network assets. An virtual device would also have been found, nevertheless the location of the physical device although has not been found, although it has been clear to some administrators that it is a physical device.
To generalise the qualitative evalutation to the personality of the test persons: The more personal motivated someone is, the more assets will be found and the more likely it is that the person discovers the real purpose of the device. The more personally interested a person is, the less effective are empty phrases and referrals to other persons and projects.

5.6 Quantitative evaluation

As well as a qualitative evaluation a quantitative evaluation needs to take place to get a better understanding of this research.
On day 1, a Wednesday, the device was planted and documented according to the companies commissioning process. An email to about this step was sent to five people, four administrators including one, who already knew about the project and one manager, who also knew about the project. Therefore three administrators who did not know anything about the project received an email about it. Two started an investigation. Another administrator started an investigation solely from the existence of a ticket, as well as a manager, the CIO who started a faltering investigation after being informed by an administrator. This results in three administrators and one manager who investigated the project. Five administrators did not start any investigation about the project, as well as no other role of Table 4.1 - DPO, documentation manager, security, operator, DevOps, Procurement or Accounting, or a passerby - raised any questions.
Two administrators started the investigation on day 1. One revealed the true purpose the same day, while the other finished investigating after three weeks without succeeding. The third administrator started the investigation on day 7, the first day after return from being out of office, as well as the manager when being informed from the third administrator. Both completed their quest the same day; the administrator revealed the true purpose, the manager only got told some empty phrases. No one started an examination after the first day, except people being out of office.
None of the persons, who have been categorised critical according to Table 4.1 have discovered the true purpose of the project, while administrators are the most dangerous staff group when trying to hide or disguise a network device.

6 Conclusion

This work showed some theoretical approaches on how to disguise and hide honey systems to personnel of an organisation to reveal an insider threat and attackers already operating from the inner network. Four key research questions have been asked, followed with some intermediate questions. The intermediate questions have already been answered in chapter 4 and confirmed in chapter 5. To get an overview of the task and answer the question “From which staff can a honey system be hidden?”, all involved personnel and assets of a network device has been listened. These lists have been used to create a matrix on who needs access to which asset and to what information level. It was concluded that there exist three groups: People who need full knowledge about a honey system, people with fake knowledge and people with absolutely no knowledge about it.

Some technical and organisational aspects (subsection 2.4.3) were mentioned which need to be in mind when hiding or disguising a honey system. To answer the question “How to hide a honey system to the inside?” it needs to be answered what exact purpose the system has, which threat it targets and which system should be used. This will lead to the question if it is recommended to hide or disguise a system. A description of various attack threats (section 4.5) and hideability (section 4.7) have been made. When comparing the hideability of devices deciding between a virtual and physical device, it needs to be noticed that administrators are the staff group which will most likely detect the device. Therefore a focus needs to be hiding the device from administrators. A small physical device is easier to hide from administrators than a virtual machine. Unfortunately, most SBC have an ARM architecture which mostly requires additional work to run community software (even with platform independent scripting languages). Also, a comparison of various team constellation has been created. When establishing this project it is essential to have a compulsory transfer routine established in case the product owner leaves the organisation. A team approach minimises the risk of a loss of control over the honey system when a specific administrator leaves the organisation.

It is recommended to only hide a system only if necessary and clues about an insider threat already exists. Hiding a network device, respectively a honey system, bears a lot of risks and is difficult to accomplish and

maintain. If possible, disguised systems should be created with a multiple team approach, therefore two or more teams maintaining honey systems only known to them. This lowers the risk of any liability problem, as any participating team gets surveyed by another team, and no feelings about injustice will rise in the teams, as this behaviour of honey systems would be a common procedure and present in the organisational culture. "Which honey systems can be hidden to the inside?" depends on the answering of the previous two key research questions. In section 4.7 a good overview was given. All considered honey systems (honeypots, honeynets, honeywalls and honeytokens) can be disguised, although honeytokens are the easiest system followed by honeynets, honeywalls, and honeypots. Hiding a system has different prerequisites and needs other precautions been made. Again, honeytokens are the easiest to hide but the other systems differ. Honeypots are easier to hide than honeynets and it is not clear if honeywalls, because of their routing behaviour, can be hidden at all. While working on this project, further organisational and legal questions arised, which strongly depend on the organisations structure and the legal landscape of the respective country. Such a project needs a extraordinary planning in advance. The device needs to be prepared before inserting to the network, as well as the documentation should already exist when inserting the device. It is recommended to test and build anything in a non-organisation related network.
This leads us to the last question "Does it make sense to hide a honey system to the inside?". Adding a hidden or disguised honey system to the network is recommended if a specific goal is defined and enough resources, as well as capable, trustworthy staff is available to perform such a project. Every organisation needs to determine themselves if the goal of such a the project is worth the money and time resources spent.

6.1 Future Work

To improve the results of this work, the experiment needs to be redone in other organisations with different staff to validate the results. Also, experiments with other honey systems in other constellations need to be conducted.
This work did not consider the existence of any fake L7 application. Also, the experiment had none implemented. Future research may include a fake L7 application and may analyse if one improves resilience to a disguised honey system and is efficient to time and money resources.
More research needs to be done to improve the storing of infrastructure code of the device. How to securely store and hide infrastructure code in a SMB or large company?

List of Figures

List of Tables

Listings

Glossary

ACL	Access Control List
AD	Active Directory
ADB	Android Debug Bridge
API	Application Programming Interface
BCM	Business-Continuity-Management
CA	Certificate Authority
CARP	Common Address Redundancy Protocol
CEO	Chief Executive Officer
CIO	Chief Information Officer
CISO	Chief Information Security Officer
CTF	Capture-the-flag
DNS	Domain Name System
DPO	Data Protection Officer
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HA	High-Availability
HL7	Health Level 7

HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICS	Industrial control system
IP	Internet Protocol
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MITM	Man-in-the-middle
MotD	Message of the Day
NDA	Non-Disclosure-Aggreement
NIC	Network interface controller
NIDS	Network Intrusion Detection System
NTP	Network Time Protocol
OS	operating system
OSI	Open Systems Interconnection
OT	Operational Technology
PKI	Public Key Infrastructure
RDP	Remote Desktop Protocol
SBC	Single Board Computer
SCADA	Supervisory Control and Data Acquisition
SIEM	Security Information and Event Management
SMB	Small and Medium Businesses
SMTP	Simple Mail Transfer Protocol

SNMP	Simple Network Management Protocol
SSH	Secure Shell
STIX	Structured Threat Information eXpression
SVM	Support-Vector-Machine
TAXII	Trusted Automated eXchange of Indicator Information
TCP/IP	Transmission Control Protocol
VCS	Version Control System
VRRP	Virtual Router Redundancy Protocol
WAF	Web Application Firewall

Bibliography

[1] B. Cheswick, "An evening with Berferd in which a cracker is lured, endured and studied," in *In Proc. Winter USENIX Conference*, USENIX, Jan. 20, 1992.

[2] Paul Lackner., "How to mock a bear: Honeypot, honeynet, honeywall & honeytoken: A survey," in *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 2: ICEIS,*, INSTICC, 2021, ISBN: 978-989-758-509-8. DOI: 10.5220/0010400001810188.

[3] R. Petrunić, "Honeytokens as active defense," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, MIPRO, 2015. DOI: 10.1109/MIPRO.2015.7160478.

[4] Niels Provos, *HoneyD: A Virtual Honeypot Daemon*, 10th DFN-CERT Workshop, 2003.

[5] Fabien Pouget, Marc Dacier, and Hervé Debar, "Honeypot, Honeynet, Honeytoken: Terminological issues," *Institut Eurécom (EURECOM), Sophia Antipolis, France, Research Report RR-03-081*, 2013.

[6] Lance Spitzner, *Honeypots: Tracking Hackers*. Addison Wesley, 2002, ISBN: 0-321-10895-7.

[7] R. Baumann and C. Plattner, "Honeypots," M.S. thesis, ETH Zürich, 2002.

[8] The Honeynet Project, *Know Your Enemy: Defining Virtual Honeynets*, 2002.

[9] ——, *Know Your Enemy: Honeynets*, 2001.

[10] Lance Spitzner, "Honeypots: Catching the Insider Threat," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, IEEE, 2003, ISBN: 0-7695-2041-3. DOI: 10.1109/CSAC.2003.1254322.

[11] David Dittrich, *Customizing ISOs and the Honeynet Project's Honeywall*, 2004.

[12] Christoph Malin, "LureBox - Using Honeytokens for Detecting Cyberattacks," M.S. thesis, UAS St. Pölten, 2017. [Online]. Available: http://phaidra.fhstp.ac.at/o:2588.

[13] B. Bowen, S. Hershkoop, A. Keromytis, and S. J. Stolfo, "Baiting Inside Attackers Using Decoy Documents," in *International Conference an Security and Privacy in Communication Systems*, SecureComm, Springer, 2009.

[14] Iyatiti Mokube and Michele Adams, "Honeypots: Concepts, approaches, and challenges," in *Proceedings of the 45th Annual Southeast Regional Conference*, ACM-SE, Winston-Salem, North Carolina, 2007, ISBN: 9781595936295. DOI: 10.1145/1233341.1233399.

[15] Abhishek Mairh, Debabrat Barik, Kanchan Verma, and Debasish Jena, "Honeypot in network security: A survey," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS, Rourkela, Odisha, India: ACM, 2011, ISBN: 9781450304641. DOI: 10.1145/1947940.1948065.

[16] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer, "Hostage - a mobile honeypot for collaborative defense," in *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN, 2014. DOI: 10.1145/2659651.2659663.

[17] E. Vasilomanolakis, S. Srinivasa, and M. Mühlhäuser, "Did you really hack a nuclear power plant? an industrial control mobile honeypot," in *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015. DOI: 10.1109/CNS.2015.7346907.

[18] Jianguo Ren, Chunming Zhang, and Qihong Hao, "A theoretical method to evaluate honeynet potency," *Future Generation Computer Systems*, 2020, ISSN: 0167-739X. DOI: 10.1016/j.future.2020.08.021.

[19] Karen A. Scarfone and Peter M. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST, Tech. Rep., 2007. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50951.

[20] Zhiqiang Wang, Gefei Li, Yaping Chi, Jianyi Zhang, Qixu Liu, Tao Yang, and Wei Zhou, "Honeynet construction based on intrusion detection," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, ser. CSAE 2019, CSAE, Sanya, China, 2019, ISBN: 9781450362948. DOI: 10.1145/3331453.3360983.

[21] W. Fan, Z. Du, M. Smith-Creasey, and D. Fernández, "Honeydoc: An efficient honeypot architecture enabling all-round design," *IEEE Journal on Selected Areas in Communications*, 2019. DOI: 10.1109/JSAC.2019.2894307.

[22] José Tomás Martínez Garre, Manuel Gil Pérez, and Antonio Ruiz-Martínez, "A novel machine learning-based approach for the detection of ssh botnet infection," *Future Generation Computer Systems*, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.09.004.

[23] L. Shi, Y. Li, T. Liu, J. Liu, B. Shan, and H. Chen, "Dynamic distributed honeypot based on blockchain," *IEEE Access*, 2019. DOI: 10.1109/ACCESS.2019.2920239.

[24] G. Chamales, "The honeywall cd-rom," *IEEE Security Privacy*, 2004, ISSN: 1558-4046. DOI: 10.1109/MSECP.2004.1281253.

[25] M. T. Qassrawi and Z. Hongli, "Deception methodology in virtual honeypots," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, NSWCTC, 2010. DOI: 10.1109/NSWCTC.2010.266.

[26] P. Lewandowski, M. Janiszewski, and A. Felkner, "Spidertrap - an innovative approach to analyze activity of internet bots on a website," *IEEE Access*, 2020. DOI: 10.1109/ACCESS.2020.3012969.

[27] S. Djanali, F. X. Arunanto, B. A. Pratomo, A. Baihaqi, H. Studiawan, and A. M. Shiddiqi, "Aggressive web application honeypot for exposing attacker's identity," in *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, ICITACEE, 2014. DOI: 10.1109/ICITACEE.2014.7065744.

[28] M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici, "Honeygen: An automated honeytokens generator," in *Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics*, IEEE ITSS, 2011. DOI: 10.1109/ISI.2011.5984063.

[29] Cristian Sandescu, Razvan Rughinis, and Grigorescu Octavian, "Hunt: Using honeytokens to understand and influence the execution of an attack," in *Proceedings of the 13th International Scientific Conference "eLearning and Software for Education"*, eLSE, Bucharest, Romania, 2017. DOI: 10.12753/2066-026X-17-075.

[30] Lance Spitzner, "Honeytokens: The other honeypot," in *Endpoint Protection*, Broadcom, 2003.

[31] Nguyen Anh Quynh and Yoshiyasu Takefuji, "Towards an invisible honeypot monitoring system," in *Australasian Conference on Information Security and Privacy*, ACISP, 2006. DOI: 10.1007/11780656_10.

[32] T. D. Wagner, "Cyber threat intelligence for "things"," in *2019 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, 2019. DOI: 10.1109/CyberSA.2019.8899384.

[33] C. C. Zou and R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance," in *International Conference on Dependable Systems and Networks (DSN'06)*, DSN, 2006. DOI: 10.1109/DSN.2006.38.

[34] Tomáš Bajtoš, Pavol Sokol, and Terézia Mézešová, "Virtual honeypots and detection of telnet botnets," in *Proceedings of the Central European Cybersecurity Conference 2018*, ser. CECC, Ljubljana, Slovenia, 2018, ISBN: 9781450365154. DOI: 10.1145/3277570.3277572.

[35] Ramneek Puri, *Bots & botnet: An overview*, SANS, 2003.

[36] Ping Wang, Lei Wu, Ryan Cunningham, and Cliff C. Zou, "Honeypot detection in advanced botnet attacks," *Int. J. Inf. Comput. Secur.*, 2010, ISSN: 1744-1765. DOI: 10.1504/IJICS.2010.031858.

[37] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos, and R. Beyah, "Rethinking the honeypot for cyber-physical systems," *IEEE Internet Computing*, 2016, ISSN: 1941-0131. DOI: 10.1109/MIC.2016.103.

[38] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling, "The nepenthes platform: An efficient approach to collect malware," RAID, 2006.

[39] Pavol Sokol, Jakub Míšek, and Martin Husák, "Honeypots and honeynets: Issues of privacy," *EURASIP Journal on Information Security*, EURASIP, 2017. DOI: 10.1186/s13635-017-0057-4.

[40] Sean Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," 2012.

[41] Julie Connolly, Mark Davidson, and Charles Schmidt, *The trusted automated exchange of indicator information (taxii)*, MITRE, 2014.

[42] Mark Davidson and Charles Schmidt, *Taxii–overview*, MITRE, 2014.

[43] *TAXII™ Version 2.1*, OASIS Open, 2020.

[44] Chun Wei Choo, Pierrette Bergeron, Brian Detlor, and Lorna Heaton, "Information culture and information use: An exploratory study of three organizations," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 5, pp. 792–804, 2008. DOI: https://doi.org/10.1002/asi.20797.

[45] L.J. Janczewski and V. Portougal, "Need-to-know principle and fuzzy security clearances modelling," *Information Management & Computer Security*, 2000. DOI: 10.1108/09685220010356247.

[46] John Hendry, "Exploring CEO identities: Leadership, accountability and role," *Accountability and Role (January 15, 2012)*, 2012. DOI: 10.2139/ssrn.2029284.

[47] Angelo Corallo, Mariangela Lazoi, and Marianna Lezzi, "Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts," *Computers in Industry*, 2020. DOI: 10.1016/j.compind.2019.103165.