Honeypots on AAU's Network

Master's Thesis

AALBORG UNIVERSITY

STUDENT REPORT

Aalborg University
Networks and Distributed Systems

[This page is intentionally left blank.]

Networks and Distributed Systems
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Honeypots on AAU's Network

Theme:
Integrating Container Based Honeypots on the University Network

Project Period:
Spring Semester 2019

Project Group:
Group 1024

Participant(s):

Rasmi Vlad Mahmoud

Supervisor(s):
Jens Myrup Pedersen

Page Numbers: 69

Date of Completion:
June 6, 2019

Abstract:

Aalborg University is having a large and complex network in order to guarantee daily operations. Due to its size and complexity is hard to assets the threats and also to determine which systems face the highest risk of being attacked. Therefore, the necessity of having system that can keep track of the events is raised.
Honeypots can mimic a large variety of protocols, therefore the latest existing honeypot technologies were investigated. Moreover, a information security risk assessment was conducted to establish the university context and to identify the groups of attackers and their motives. Since is desired to integrated the honeypots with Aalborg University's network its structure was analyzed in order to determine honeypots' position that can give the insights into system probed.
To integrate the honeypots with the existing AAU's network, Docker engine was installed on a virtual private server provided by the university. The subnet used for the honeypots is located outside the university firewall for the traffic to not be affected. Every container is assigned an static IP address from the available that has access to/from the Internet, additionally honeypots are capable of tracking the interactions in individual log files that can be further analyzed. Nonetheless, honeypots being active on AAU's network they act as a decoy system by attracting the attackers.

[This page is intentionally left blank.]

Preface

About

The report is written and completed as master's thesis for MSc. of Networks and Distributed Systems at Aalborg University.

The theme for the project is "Integrating Container Based Honeypots on the University Network".

Report reading guide

All the referenced sources for the report can be seen in the bibliography. For the source references the Vancouver method is used. I.e. a source will be referenced with the following method and layout, with numbers and square brackets:

> "Example text [1]."

Where the source referenced is number "1" in the bibliography.
Sections and chapters are referenced with chapter and section numbers followed by one another:

> "Example text (see section 2.3.1)."

Where the section referenced in the text is chapter 2 section 3 and subsection 1.

[This page is intentionally left blank.]

Acronyms

AAU Aalborg University.

BYOD Bring-Your-Own-Device.

FTP File Transfer Protocol.

GELF Graylog Extended Log Format.

HIHP High-Interaction Honeypots.

HPC High-performance computing.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ICS Industrial control systems.

IMAP Internet Message Access Protocol.

IPS Intrusion Prevention System.

JSON JavaScript Object Notation.

LIHP Low-Interaction Honeypots.

MIHP Medium-Interaction Honeypots.

MPI Message Passing Interface.

MQTT Message Queuing Telemetry Transport.

MSSQL Microsoft SQL Server.

NIC Network Interface Card.

NIDS Network Intrusion Detection System.

OS Operating System.

POP3 Post Office Protocol 3.

POP3S Post Office Protocol 3 Secure.

RDP Remote Desktop Protocol.

SIP Session Initiation Protocol.

SMB Server Message Block.

SMTP Simple Mail Transfer Protocol.

SSH Secure Shell.

TFTP Trivial File Transfer Protocol.

TTPs Tactics, Techniques and Procedures.

UI User-Interface.

UPnP Universal Plug and Play.

VM Virtual Machine.

VMM Virtual Machine Monitor.

VoIP Voice over Internet Protocol.

VPS Virtual Private Server.

Table of contents

CHAPTER 1

Introduction

Cyber security is a fundamental requirement for all the organizations,and this is due to the tremendous growth of information and networks. Every institution has particular cyber security challenges to deal with, and there are different approaches and perspectives that are used for evaluating the risk and to manage the security across institutions.

Educational and research institutions are dealing with specific cyber security threats due to their huge amounts of digital data, used both for the normal operation of the organization, but also what is tagged as research data. For instance, universities carry on large variety of activities and they cannot have the typical organizational boundaries, therefore universities are required to establish custom security policies to accustom their needs. [32]

In the report from 2018 Cisco is focusing the public sector, and furthermore education receives special attention and share insights from schools, high schools and universities from United States. Over half of higher education (58%), reported that they have had experienced at least one security breach, and this percentage is the highest over all public sector industries. This types of breaches are most of the time identified with damage to the institution reputation, additionally nearly 51% of the attacks resulted into lost of money, over 500,000$, for the universities . [6]

Besides, Danish Defence Intelligence Service Centre for Cyber Security (CFCS) stated in February 2017 that foreign states are conducting acts of espionage against Danish research. The curiosity is generated both from political and commercial motives, but nonetheless assailants were interested also in the institution infrastructure that can be used for attacking other public Danish institutions. [13]

Additionally, in 2018 Cisco's report 41% of the universities claimed that budget limitations and lack of trained personal (27%) are some of the major impediments that they are facing. Moreover, universities report that they have employed half of the median number of security personal that most of the enterprises have. This insufficient number of personnel results in reduced threats' investigation and deployment of new technologies that can aid to make their

security posture more powerful.[6]

Huge amount of data require a method to alert and monitor the unauthorized access to network. Traditional firewalls are used to filter the inbound and outbound traffic, but this method is not able of informing who is trying to breach the network. After, there are Network Intrusion Detection System (NIDS) and Intrusion Prevention System (IPS) that are mechanism used for prevention and detection, additionally another method of protection is to have a decoy system that lures that attacker into thinking that he is actually on real system, while he is trapped and monitored inside a Honeypot. [2]

Honeypots are systems that attract attackers into trying to breach them. This interest is generated by the idea that a honeypot is looking like a real system in the network. Furthermore, these systems are voluntarily created to be attacked or compromised.

A honeypot permits incoming attacks and offers the possibility of monitoring and collecting information while acting as a decoy system. Honeypots are usually designed for catching any actions that an actor is performing when an attack is initiated. By offering these possibilities honeypots show great value when dealing with zero-days exploits.

1.1 Contribution

The master's thesis is investigating into the possibility of deploying honeypots within a large and complex network. Honeypots permit a high flexibility in terms of design due to their possibility of integration with a physical or virtualized environment. Furthermore, a systematic approach is presented to integrate and deploy honeypots into a corporate network. Docker containers are configured to act as real device on the network, by assigning a static public IP for the containers that have access to/from the Internet. The subnet where the containers are configured, is a subnetwork of the Danish Research Network, consequently every container will be hosting a different honeypot that permits access using the configured static ip address.

1.2 Report Structure Outline

This section is presenting the report overview focusing on the structural organization, therefore in the followings a summary of the chapters can be seen:

Chapter 2 - Problem Analysis
Starting by presenting honeypots technologies according to the state of the art, further risk assessment is performed for a better understanding of the university perspective. Accordingly, the problem statement is formulated, and after that Aalborg University network structure is investigated. At the end, a rough idea of the desired system is discuss

Chapter 3 - System Requirements
Focusing on the system requirements that are used to describe system, but also on system attributes. At the end, development concerns are discussed.

Chapter 4 - System Design
Starting by analyzing the possibilities of hosting environment for the honeypots, and chapter is continuing by a rough presentation of Docker technologies that have been used. Later, the system architecture is outlined, by focusing on the main components. Chapter is ending by discussing how the logs are processed.

Chapter 5 - Docker Security
Focused is put into the system's aspects in terms of security, by detailing docker security features and presenting the firewall settings that were used to isolate the honeypots.

Chapter 6 - Honeypots Deployment
Architecture setup is presented by focusing on two different angles: the attacker's perspective and the defender one. At the end, details are given for processing and organizing the stored log files.

Chapter 7 - Results and Observations
Overview of the honeypots activity is presenting by considering at the beginning general aspects. Further, the honeypots are discussed one by one.

Chapter 8 - Conclusion
Final observations are outlined by considering the main ideas of the report. Nonetheless, final answers for the research questions are provided.

[This page is intentionally left blank.]

CHAPTER 2

Problem analysis

The chapter is starting by presenting the state of art honeypots technologies, to create an initial overview of the possibilities. Further, a risk assessment is performed to determine the university context, from a structural perspective the chapter is organized as follows:

- Overview of State of Art - general aspects and honeypots technologies are detailed
- Overview of the Risk Assessment - conducted to identify the threats for AAU and also established the university context
- Overview of the Problem Statement - is identifying the problem and also formulating the research question
- Overview of AAU's Network - is presenting the network structure to understand some its main components
- Overview of Desired System - is presenting an initial sketch of the system without going in any technical details.

2.1 State of the Art

Honeypots can be classifyed based on certain criteria. For instance, based on the level of interaction, which is determined based available commands that an attacker is having access to when is inside a honeypot and also on received feedback, honeypots can be classified in: Low-Interaction Honeypots (LIHP), Medium-Interaction Honeypots (MIHP) and High-Interaction Honeypots (HIHP).

LIHP

LIHP offers a limited range of service for the attacker to use. There is no operating system for the attacker to use, but this type of honeypot can emulated service like Secure Shell (SSH) or File Transfer Protocol (FTP) that can be attractive for the assailant. Its main advantages are represented by the aspects of being easy to deploy and maintain, furthermore these types of honeypots are excellent statistical tools by being able to detect high peaks of requests. Although, their drawbacks are the facts that they are limited in discovering new attack patterns and their response is moderate to exploits. [23]

MIHP

In contrast with LIHP, MIHPs offer more interaction and access to the attacker, although no operating system is available. The emulated services are more complex and are able return a response that can attract an attacker to carry on with the attack. Due to the level of interaction and availability of commands, both MIHP and LIHP, are having low changes of being compromised. [23]

HIHP

By far the most advanced type of honeypots. These honeypots offer the attacker a real operating system to use that is not restricted, therefore require more work for deployment and maintenance. Accordingly the complexity these honeypots offer a large variety of monitoring services, including attacks logs, data access, file traversing and so on. Nonetheless, due to the high amount of information generated the analysis needs to be done manually. While, the first two did not have a high risk of being compromised, this type of honeypot present a high risk and it needs to be monitored permanently [23]. In table 2.1 a summarize of honeypots types and their primarily feature can be seen.

	LIHP	MIHP	HIHP
real operating system	no	no	yes
risk of compromise	low	mid	high
wish of compromise	no	no	yes
information gathering	low	mid	high
knowledge to deploy	low	mid	high
knowledge to develop	low	high	high
maintenance time	low	low	very high

Table 2.1: The table provide in [23] is summing up the principal characteristics of LIHP, MIHP and HIHP

In table 2.2 a classification of the existing honeypots is started. The honeypots will be categorized into a scale of *Low*, *Medium* and *High* based on their functionalities. For instance, *Information* is considerate the amount of data that the honeypot is capable of tracking and for *Risk* is considered if the honeypot can be compromised based on their commands availability. Therefore, for an LIHP honeypot the risk of compromise is *low*, since is not a real operating system. Consequently, HIHP will be tagged as *high* for the information column since is able to provide the most information about the attacker and it's Tactics, Techniques and Procedures (TTPs).

Honeypot	**Information**	**Risk**
Cowrie	Medium	Medium
Conpot	Low	Low
Kojoney2	Medium	Low
Glastopf	Medium	Low
Shockpot	Medium	Low
Thug	Medium	Low
Modern Honey Network	High	Medium
Dionaea	Medium	Low

Table 2.2: Existing honeypots frameworks classification.

Moreover, honeypots can be classified based on their desired purpose. For instance, there are Research and Production honeypots, in the following the characteristic of each type will be further detailed.

- Research honeypots - are basically used to gain information about the attacks and attacker. The research honeypots are used mostly by educational institutions, military or government organizations with the role of gathering as much information as possible about TTPs. Beside, their advantages these types of honeypots have also some drawbacks: they are not offering direct value to the organization and are hard to maintain, but the information that they offer is extremely important. Using information, organizations can develop new policies to stay ahead of the cyber threats. [23]

- Production honeypots - are primary used by companies and are easy to deploy and maintain. These honeypots are placed inside the production network to improve overall security. These honeypots are mimicking the production network, by offering services such as FTP, Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP). They guard the principal network by deceiving the attacker, and by alerting the network's administrators of the activity. Even if the amount of information provided is less, the risk of being compromised is lower. [23]

There are a variety of existing honeypot frameworks, but some of them are already outdated by this time and therefore the research will be focusing only on the those that were developed starting with the year 2015. The purpose is to review some of the existing frameworks that are still maintained and also to chose the most appropriate ones for the project based on the section 2.2 and the chapter 3 that will be discussed further in the report.

Cowrie is a MIHP capable of logging brute force attacks on SSH and Telnet with the capacity of keeping track of the interactions an attacker is having with the shell. In addition, Cowrie is having a fake file system with the ability of adding or removing files, and also all the downloaded files are saved for later inspection. [23]

Conpot is a LIHP of honeypot that aims to collect intelligence about the attackers that are targeting systems that monitor or control industrial processes. Conpot is able to understand common Internet protocols such as HTTP, but also some Industrial control systems (ICS) specific ones like kamstrup, BACnet or mosbus. [23]

Kojoney2 is a MIHP that is listening on port 22 for incoming SSH attacks. The same like Cowrie is capable of downloading the requested files and place them into secure locations, furthermore its purpose is to fingerprint attacker's behavior and tools. When is exposed to external network, Kojoney2, can identify attacks' source and well as fingerprinting the attackers' moves after he gets access.

Glastopf is a modern LIHP web server. This type of honeypot gathers information about web-based attacks such as local and remote file inclusions or SQL injections. Moreover, Glastopf is capable of downloading files that are requested, keeping the attacker's interest to continue further with his attack.

Shockpot was created after the vulnerability Shellshock/CVE-2014-6271 of the bash shell. It is emulating an Apache-server that allows attackers to execute arbitrary code into the original vulnerability.[23]

Thug is a LIHP that emulates a real web-browser actions and its main task is to detect malicious code. Thug uses Google V8 Javascript engine that is embedded with pyv8 for analyzing malicious code and Libemu library with Pylibemu for detecting a shellcode emulation.[35]

Dionaea is a honeypot that offers exploitable network services. Some of the offered protocols are: Server Message Block (SMB)SMB, HTTP, FTP, Microsoft SQL Server (MSSQL), or Voice over Internet Protocol (VoIP). Its action is to trap malware that attacks the stated protocols and furthermore to obtain a valid copy of it.

Modern Honey Network is a management system capable of integrating some of the previously presented honeypots sensor such as: Cowrie, Dionaea, and Glastopf, and others. MHN is collecting the attacks' data and is able to plot the data into a World Map view while maintaining all the information about the attack parameters.

As an overview, there are different types of honeypots capable of mimicking various types of protocols and systems. Moreover, some of the honeypots require higher maintenance time then others, but also the information provided is broader. Therefore, it is important to consider not only the data that honeypot is producing, but also the required time deployment, maintenance and its applications area.

For outlining the types of information a honeypot can provide, two long term honeypot project

that are still active and maintained will be discussed. These projects are both used as monitoring tool, but also as a prevention and detection ones. They provide important insights into the number of attacks, sources and destinations of the attacks.

The Finish security company *F-Secure* have released a report [12] about 2018 attacks landscape. Their honeypots' network revealed that U.K outranked Russia in 2018 at the total number of attacks, and that U.S is still the preferred country when comes to being a target. When it comes to which country is haunting which, F-Secure have stated that Russia is after U.S, but also after U.K, Ireland and Netherlands. In fig. 2.1 an overview of sources and destination for the attacks can be seen. Moreover in fig. 2.2 a display of the number of the attacks and port that was probed is presented. It can be seen that the highest activity was on port 445.

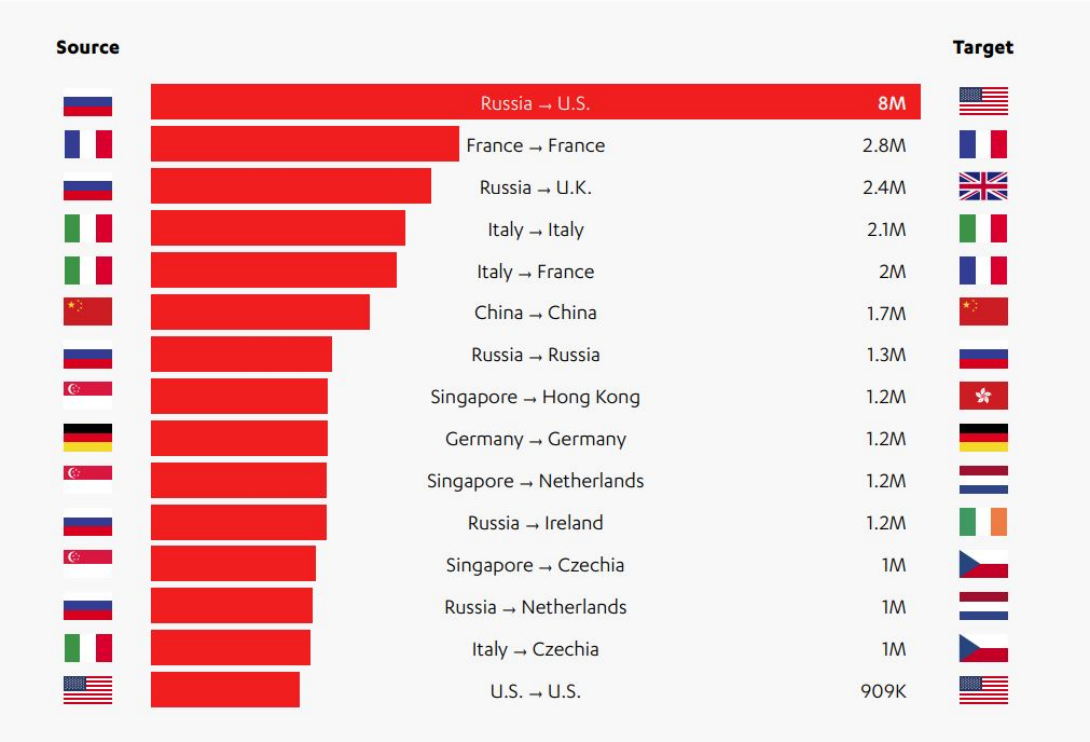


Figure 2.1: Most relevant relation source-destination provided in F-Secure report[12].

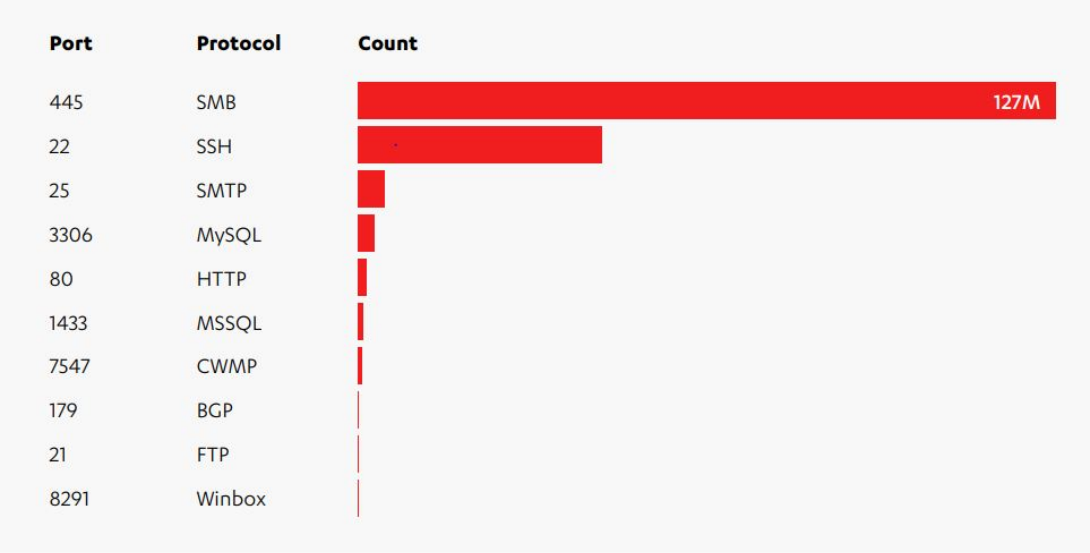


Figure 2.2: Top TCP ports probed provided in F-Secure report[12].

Additionally, the German company Telekom have launched in 2013 a project where 97 honeypot's sensors where deployed over the world and the project is still going and providing insights of the attacks. A real-time cyber attack map based on the data collected by the honeypots is presented on the site *https://sicherheitstacho.eu/start/main*. Telekom uses the information gathered to protect their own systems and shares the data also with security vendors. On the website there are available statistics about total number of attacks, type of sensors that are used and the protocols used in probing [11]. For instance, in table 2.3 an overview of the top countries and their number of attacks for March 2019 can be seen.

Country	Attacks
Russia	59011166
United States	46167303
Poland	28393206
China	16620060

Table 2.3: Overview of the top ranked countries by number of attacks. [11]

The following section will present an analytical method in identifying information security risk for AAU. The university is having a huge amount of data and also a big variety of systems, it is valuable to established the cyber security context that AAU fits into and also to form a general overview over the possible threats and their principals groups that can perform cyber attacks. Nonetheless, performing a risk assessment is a method for AAU to identify risks before any actual problems are appearing.

2.2 Risk Assessment

The ISO 27005 Information Security Risk Management standard was used in order to have a systematical approach towards the organization's risk. Not to mention, that risk assessment represents a crucial analysis within any organization and based on it the whole core of information security management is established [28]. Furthermore, is one of the principal methods that

organization uses in order to determine what related technologies are best for protecting their information assets [28]. The risk assessment process can be divided into three distinct branches that are outlined in the fig. 2.3.

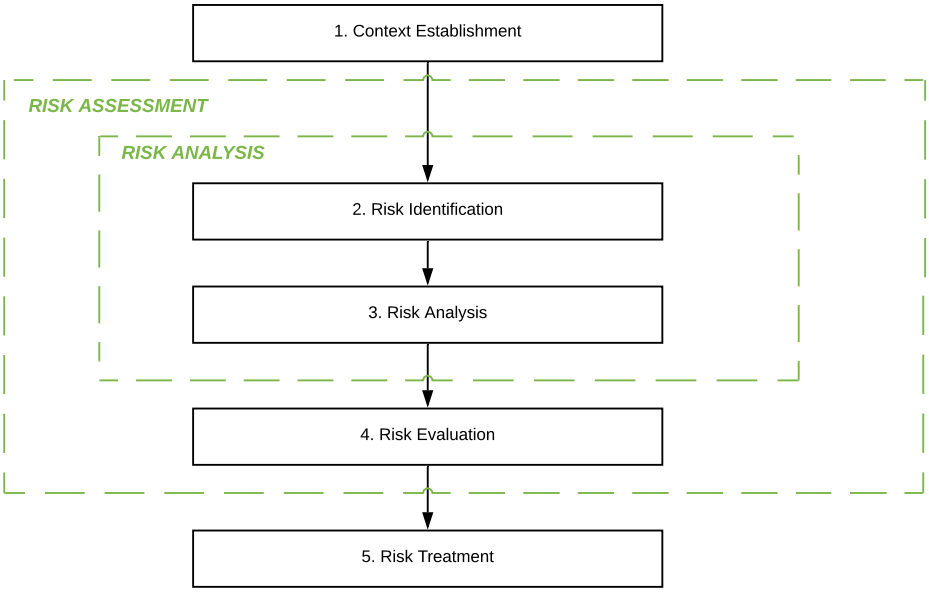


Figure 2.3: ISO 27015 steps in performing a risk assessment.

2.2.1 Context Establishment

Universities are special case of organizations due to their high amount of data, large number of users, but also their special policies when it comes to cybersecurity. The type of information universities posses make them important targets for the attackers. University are not only having personal information about their employees, but they also hold valuable research data. Nonetheless, Bring-Your-Own-Device (BYOD) policy is making it even more difficult, since student's devices can have outdated software and this is creating proper channels for the attackers. In the following subsections, AAU context will be established from the perspective of a research institution, an educational one and also as organizational one.

Research Context

The research that Aalborg University holds make it a valuable target for foreign intelligence services. AAU is targeted not only due to its research information, but also due to its third party data that is used to perform the research. Third parties' information can come in different forms (medical/health records, cutting edge technologies) or detailed information about discoveries. AAU stands in vulnerable position since not only it needs to proper store the data, but it also needs to protected it. In the event of threat which can alter the data, AAU can be held responsible and this is implying serious damage to brand. Its reputation can be diminish by losing its credibility and nonetheless, the circumstances can escalate to legal action.

Educational Context

Daily hundreds of students are relying on AAU's network to access their educational material, moreover the students at AAU are having access to different systems that can aid them in their studies. Nonetheless, daily personal students' devices connect to the university networks which makes it nearly impossible to control their security state. In addition, the university holds detailed personal information of their students such as CPR. numbers, home addresses, even electronically copies of students' passports. Therefore, this data can be priceless for any attackers that can gain a lot of money selling it on the black market.

Institutional Context

Additionally from the students at AAU there are approximatively 3500 employees. The necessary equipments (laptop, work stations, smart devices) are provided by the university, but it still comes down to the personal information that the university is holding about their employees. Information such as contracts, bank statements, medical records and CPR numbers can be considered at risk in the event of a threat.

2.2.2 Risk Identification

This subsection will be focusing on detecting the risk that can harm the assets. By asset in this analysis is understood any person, equipment, or third party that is involved in normal function of the university and presents a threat of being attacked. Therefore, the analysis is made by first defining the general type of assets and then the possible attacks' groups that might have the necessary resources and motives.

Hardware - Any systems that are used during a normal work day at the university. For instance laboratory equipment, employees workstation/laptops or servers

Software - Programs, applications, websites that were developed for and by AAU to perform its daily activities. For example some of the software assets for AAU can be Moodle, STADS, RES, RUS or ESDH. More detail about the systems can be seen in table 2.4.

Information - at this category can fit any data that the university is holding, even if is employee, students or research data

People - Employees expressed by academic staff, researchers or any other personnel.

Outsourced Services - in this category can fit the third-party services that are either bought or lend to the university for research actions.

System	Description
Moodle	Moodle is a virtual learning enviroment that AAU uses, where courses material/information are posted. Uptime is essential for the AAU to operate as a educational institution
STADS	STADS is the study administration system that handles courses sign-up and grades access. Therefore, concerns about grades falsification are raised, but also personal data makes it a critical system.
Email service	Email service represents the main form of communication at AAU. Emails can contain sensitive/confidential information, therefore an attacker can potentially disrupt the normal state by removing the spam filter and letting accounts exposed.
Budgeting and Planning Tool (RES)	RES is a budgeting and planning tool for resource management. The RES is an important tool for key management leaders for making qualified and long-term decisions. Therefore, if is attacked can offer to the attackers a lot of information.
Economic system	If an assailant is getting access to the economic system, data can be altered. Bank statements or accounts can be modified resulting into transaction being made to the attacker's account.
Travel / outlay system (RUS)	RUS is the system used to manage the employees travels. Therefore if it is exploited by attackers, they can benefit of free expenses and also gain information about employees travelling plan.
Electronic Case and Document Management (ESDH)	Is the system that is handling all the documents at AAU. Compromising this system can lead to legal and reputation damage due to wide type of information the system can poses.

Table 2.4: Overview of the systems that are used at AAU

Once the assets' list was put in place, the next phase of identifying the source of threats can be started. This step along with the next ones are important, because by narrowing down who can possible attack the university is helping to understand which are the possible threats and vulnerabilities. Therefore, some of the well-known groups that have the necessary resources to attack the university's assets are presented in the followings:

- **Script Kiddie** - their motivation comes from the desire of proving himself among others, can show strong persistence even if he is facing failures. They are not skilled enough to perform complicated attacks and their limited in terms of computational and financial resources. [29]
- **Cyber-terrorists** - are motivated groups to cause harm in the society by attacking the critical infrastructure. They have strong ideological and political motives, also they might go to extreme. Their resources are significant and are capable of planning long attacks that require preparation and carrying out. [29]
- **Black hat hacker** - his main motive is financial gain. His skill is varying, but is worth considering that some of them are cybersecurity world's experts. If they are organized into a criminal structure their resources can be vast both computational and skill matter. [29]
- **Hacktivist** - highly similar with the cyber-terrorist groups, with the main differences that hacktivists are not going to extreme and they focus their potential on selected groups, politicians or individuals. Their resources and competences can be varying, since most of them are acting solidarity. Once they are organized their computational resources and competence are rising. [29]

- **Insider** - in this category is falling any untrustworthy employee who is motivated by financial gain and also to cause the employer harm. In terms of resources, he might have access to all the systems and have deep information about the organizational structure and its security. [29]

A threat to an assets can be understood any *"Potential cause of an unwanted incident which may result in harm to a system or organization"* [29]. Classifying threats into categories based on cause and principal actors the following classes can be determined:

Natural - power failure, floods, fire, earth quakes

Unintentional - a non-hostile actor, either an insider or an outsider

Intentional - outsider/insider hostile individual or group (organized hackers groups, foreign agencies, industrial espionage, terrorists), or non-hostile that his principal motive is curiosity

Since, the natural types of threats are not interesting for this report scope, further detail will be put into the unintentional and intentional classes. Therefore, a non-hostile actor can be an employee that misuses some of the systems that he has access to, and even if the action is done purely accidental this can still cause loss of data, system malfunction or alter of the available information. In the following table 2.5 threats are imagined based on their assets and principal actors.

Group/Threat Source	Assets	Threats	Attack Methods
Script Kiddies	Hardware Software Information	Not responding servers due to DDoS Services as Moodle,STATS,RES are not working due to attacks Sensitive information are obtain from servers Weak credential are exposed	DDoS Attack SQLi XSS Dictionary attack Brute-force attack
Black hat hacker	All-above mentioned People Outsourced Services	All-above mentioned Employees workstation blocked for ransom Access to internal systems Malware propagation by email	All-above mentioned Phishing Spear-phishing Trojans Ransomwares
Hacktivist	Software People Outsourced Services	Not responding servers due to DDoS Expose of sensitive research information	Social-engineering Phishing Spear-phishing Malwares DDoS
Cyber-espionage	Hw/Sw Information People Outsourced	Not responding servers due to DDoS Research information are captured	All-above mentioned AI-powered attacks
Insider	Hardware Software Information	Intentionally shutting down security systems Copy sensitive information from servers Exposed credential to systems Trade brand secrets	Trojans MITM DDos

Table 2.5: Threats linked with the assets and their principal actors.

Continuously, another determinant factor at risk identification are the vulnerabilities. Threats actually are taking advantage of them for causing harm. Precisely, *A vulnerability is a weakness, flaw, or deficiency that can be exploited by a threat to cause harm to an asset* [29]. Vulnerabilities

can be categorized in different classes: Phishical, Hardware/Software or Human vulnerabilities. Therefore, in table 2.6 a relation between vulnerability type and assets is developed, in order to expose the defenseless actions.

Vulnerability class	Assest	Vulnerabilities
Hw/Sw	Hardware Software Information Outsourced services	Old O.S. Software outdated Lack of back up servers Weak encryption and integrity check Service stop running and Erros
Phishical	Hardware Information Outsourced services	Unsecure hardware plug-in Phishical remove of device
Human	People Information	Social-engineering Bring data out of the office Store data into personal devices Lack of security Awareness Sensitive information discussed outside the office Misuse of the systems

Table 2.6: Relation between the vulnerability type and corresponded asset is presented.

2.2.3 Risk Analysis

Based on the previously established assets, threats and vulnerabilities the actual risk needs to be determined by judging the frequency of threats and the severity of exploited vulnerabilities. Since there are not available previous statistics of the attacks the following analysis will be made from an visionary point of view. Threat's likelihood based on the attacked assets and consequence is discovered, moreover it is classified into *Likely*, *Possible* and *Rare* .

In addition, impact is chose between *Low, Medium or High* determined by the possible damage and the threat probability of occurrence. Threats are based on the vulnerabilities that are discussed in table 2.7. Furthermore, in table 2.8 these facts are outlined and the consequence is explained:

Vulnerabilities	Severity	Explanation
Old O.S.	High	Old operation systems have vulnerabilities and those are exploited by attackers
Software outdated	Medium	Antivirus
Lack of back up servers	Medium	Temporary out of service systems Loss of data
Service stop running and Erros	Low	Short/long time annoyance
Unsecure hardware plug-in	High	USB with malicious server is insert into an important system
Phishical remove of device	High	System containing sensitive data is stolen
Social-engineering	Medium	Employee can be tricked into divulging company information
Bring data out of the office	Medium	Unauthorized person can read, steal them
Store data into personal devices	Medium	Personal device can be infected with malware
Lack of security Awareness	High	Employee are not coping with the security details. As consequence no interest is showing in good cybersecurity practice.
Sensitive information discussed outside the office	Low	Outsiders might gain company information

Table 2.7: Vulnerabilities and their impact severity.

Threat	**Likelihood**	**Impact**	**Comments**
Not responding servers due to DDoS	Likely	Medium	Depending on the actual server that is attacked, the impact can vary from Medium to High
Services as Moodle,STATS,RES are not working due to attacks	Rare	High	Being some of the most used services at AAU, an off period can results in high disturbance
Sensitive information are obtain from servers	Possible	High	AAU is holding both research and personal information, if any of previous are tampered AAU can be held responsible
Weak credential are exposed	Likely	Low	Depending on the exposed credential and their associated systems, the impact can vary
Employees workstation blocked for ransom	Possible	Medium	Employees workstation may held sensitive information, that can be alter
Access to internal systems	Possible	High	Research of personal data can be tampered
Malware propagation	Likely	High	Not depending on the malware type, if the network is infected can have dangerous consequences
Expose of sensitive research information	Possible	High	As sensitive information can be classified patient records, shared technologies from third party or secret project documentation
Research information are captured	Likely	Medium	The dependency can become crucial if data is marketed
Intentionally shutting down security systems	Likely	Medium	The action can facilitate an attack and therefore the impact can be varying from
Copy sensitive information from servers	Likely	Medium	Depending on the actual information and if their are markted, the impact can raise to High
Exposed credential to systems	Possible	High	System access, implicitly means access to valuable data
Trade brand secrets	Possible	Medium	Operational and security policies can be exposed by rouge employes

Table 2.8: Previously presented threats and their Likelihood of occurrence at AAU.

2.2.4 Risk Evaluation

The risks were identified and their consequences and likelihoods were analyzed. Therefore, the following step is to arrange the risks based on the based on its annoyance level. Moreover, risk can be also mapped base on its probability of occurrence and impact level in order to determine the organization tolerance line.

2.3 Problem Statement

It has been seen during the risk assessment moments when the situation were based on other universities research, due to the fact that other information were not publicly available. Moreover, another aspect that came uncertain and difficult during the risk assessment was a method to classify the most critical targets for an organization with various system and labor personal. Additionally, was complicated to determine how to better dedicate the available resources.

Therefore, honeypots become suitable systems due to their large applicability that can also stand as a decoy/alert system, but also as information gathering systems. Accordingly, the following question is raising:

"How can a honeypot network be integrated with AAU's network, that could serve both as decoy system but also as a information gathering tool?"

In order to integrated a new system into AAU's network a good understanding of the actual structure of the network is necessary. Therefore, in the next section AAU's Network will be presented based on knowledge that was gained from a previous student project. It is worth mentioning that the mapping of the network was made from an insider point of view where all the system were visible.

2.4 AAU's Network

In previous project Aalborg University network was determined using NMAP and by trace-routing different connection betweens buildings. As a starting point for the analysis an existing documentation [31] was used and also additional insights were provided by the AAU IT-department. An overview of the AAU's Network map can be seen in fig. 2.4.

All incoming and outgoing traffic to/from AAU is going through *130.226.249.121* and *130.226.249.122*. Then, incoming traffic passes through an edge into one core. It should be noted that occasionally, it can be seen in the trace-route that two cores are passed through. These two cores are connected together due to routing and to ensure redundancy. However, they act as one core.

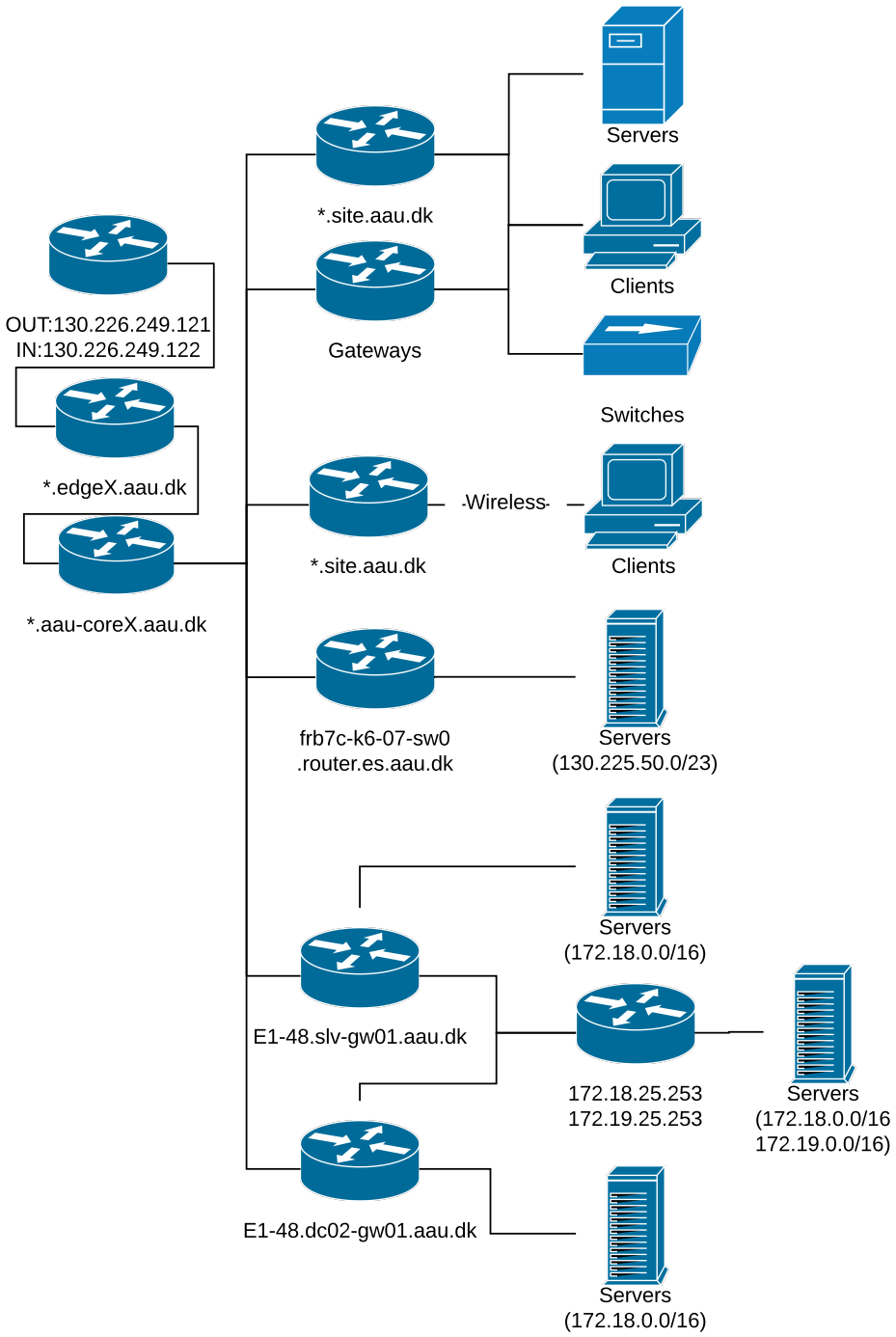


Figure 2.4: Overview of AAU network map.

After the incoming traffic has passed through the core, destination is either to a *site* or a *gateway*. From these sites and gateways, it goes to clients, servers, or switches. When performing a trace-route between two systems on AAU's network, it normally have to pass through the core before it can reach its destination. The exception is if the two sites are sharing the same router. Then they can jump directly from one site to another. Furthermore, the servers are primarily located behind *frb7c-k6-07-sw0.router.es.aau.dk*, *E1-48.slv-gw01.aau.dk*, or *E1-48.dc02-gw01.aau.dk*. However, some are actually located a level deeper behind the last two gateways, precisely behind *172.18.25.253* or *172.19.25.253*. The subnet for the servers is *172.18.0.0/16*, *172.19.0.0/16*, and *130.225.50.0/23*.

AAU has complex network that is promoting both security and redundancy, implicitly these supports the network high availability that is required for AAU's services to operate at normal parameters. Therefore, if the honeypots are chosen to be deployed outside the university firewall the subnet should originate from **.edgeX.aau.dk*. Consequently, this position facilitates the honeypots to receive attacks from outside world without the traffic being filter by the university's firewall.

Moving towards a system sketch based on the question raised in section 2.3 and the difficulties that were experienced during the risk assessment, next section will describe a desirable system formulated in natural language without integrating any technical concepts or following a well known methodology.

2.5 Desired System

Based on the question that was raised in section 2.3, the difficulties that were encountered during the risk assessment, table 2.5 and table 2.6, and due to the shortage of information supposition was used. Therefore, a system that can give any information in relation with the possible attacks that AAU can experience is wanted. Moreover, the system is required to be integrated with AAU's network in order to make it attractable for possible attackers. Nonetheless, this system is expected to be attacked and proper isolation from AAU's systems is necessary, since is not desired to be used as jump station for attacking the university services.

Additionally, the system may be considered a decoy from the AAU's network, but also as a tool that is able to gather information about the attackers. Apart from integration part, the honeypots need to have at least a limited response to attacker' request for keeping the interest active and to exclude risk of being discovered.

Furthermore, the hardware used in the system deployment is an important choice, since there are a variety of existing ones. Is desired that the system to be handled into an environment separated from AAU network, without naming any restrictions to a physical or virtual environment. Additionally, the system is required to be connected to Internet for permitting people to connect to the system. Nevertheless, the configuration should not present a risk for university systems or users, and the system is not considered a critical one where the uptime is vital, but is desired to have an acceptable work time.

The following chapter will translated the properties into more technical concepts. Moreover, based on the properties system requirements will be presented together with setup requirement and the system attributes in terms of architecture for long term solution.

[This page is intentionally left blank.]

CHAPTER 3

System requirements

This chapter purpose is to translate from section 2.5, section 2.3 and the analysis made in section 2.2 into system requirements and to define options that are worth analyzing for hosting the environment. From a structural perspective the chapter is organized as follows:

- Overview of the systems requirements - determine the system function based on the analysis
- Overview of the setup concerns - are presenting some of the setup requirements
- Overview of the system attribute - is discussing the main attributes the system should have

3.1 System Requirements

The system needs to fulfill different purposes and therefore the requirements section will be organized based on objectives. Firstly the systems needs to act as a decoy system for the real network and the information related with the attacks should be stored for further analysis. Accordingly, the following system requirements are detached:

3.1.1 SR1 - Configurable and Portable System

The desire is to have a system that can be easily configured on existing platforms and where the necessity of buying new hardware is not present. Moreover, is desired that the implementation to be rapidly movable between systems without investing new time in configurations and settings. Nonetheless, platform independence is a considered a plus.

3.1.2 SR2 - Integration with AAU's Network

Based on the facts that were discussed in section 2.2.1, the honeypots need to be associated with AAU's network. Honeypots should appear as active systems when an IP range reserved for Danish Research Network is scanned. Nonetheless, honeypots need to appear as part of AAU's network.

3.1.3 SR3 - Mimic Systems and Protocols used at AAU

The honeypots are required to imitate common services from AAU that were discussed in table 2.4 and in section 2.2.2. The system should have at least one honeypot that is able to emulate a server, a web server activity or a mail-server.

3.1.4 SR4 - Low/Medium Response to Requests

The systems require not to be detected by the attackers as honeypots, and to act as real systems to keep the attacker engaged a longer time to extract information. Thus, in order to keep attackers motivated the honeypots needs to offer a feedback to attacker's actions.

3.1.5 SR5 - Register Connections

Honeypots can be also used as alert systems, idea that was previously considered in section 2.3. When a connection is initiated, information of that instance is registered into log files. Distinct logs file will be kept depending on the used honeypots.

3.1.6 SR6 - Save Information in Relation with Connections

Most of the information that was presented in section 2.2.3, table 2.8 were based on other universities' reports, since no available AAU data was found. Therefore, after the connection is registered, related information are stored into the honeypot log files for further analysis.

3.1.7 SR7 - Monitor the Connections

Consequently, from SR6, section 2.2.3, table 2.8 and table 2.7 the system should be able to keep track of the interactions that an attacker is having with the systems. Honeypots are required to save the services that were probed and to show an overview based on ip, port and protocol attacked.

3.1.8 SR8 - UI for Statistics

Based on the problem statement that was discussed in section 2.3 the system should be use as a information gathering tool, and therefore a method to visualized an overview of honeypots interactions is necessary. Moreover, a combination of *user-passwords* used by attackers should be displayed and also most active countries based on the IP address.

Accordingly, the following system's requirements that will be formulated can also provide insights into the attacks, but due to time constrains will not be analyzed in this project.

3.1.9 SR9 - Extract Common Patterns

During the risk analysis, section 2.2.3, some of vulnerabilities and threats from tables table 2.7 and table 2.8 can benefit from further analysis. It is worth compiling a method that can extract the first commands that are tested after access is gained to a system.

3.1.10 SR10 - Track the Connections per Day

It is interesting to see if the ratio between number of connection/per day is different in the first days when the honeypots are deployed comparatively with a week later.

The following section will be investigating into the setup details, by focusing into the options that will be used for deployment.

3.2 Setup Concerns

For deploying a system, consequently a platform is needed to sustain its capabilities. When it comes to honeypots there are some possibilities that can fulfill this requirement. Furthermore, since the honeypots will be deployed on AAU's network the level of details will be restrained to physical or virtual systems deployment.

Physical Hardware

For this type of setup, an actual machine that will work as a server for the honeypots is necessary. The physical machine is required to have a screen attached or to be accessible over a SSH connection from another computer. Additionally, the computational power required is one that can assure normal operation of the system.

Virtual Hardware

Honeypots can be also developed inside a virtualized environment that acts like a physical machine. For instance, virtual boxes, virtual machines or docker's containers can represent solution for this type of setup.

Internet Connection

Nonetheless, for the honeypot to be functional a constant Internet connection is required. This can either be achieve by a direct connection for the physical setup, or if a virtualized environment is chosen the hosting machine for the environment is required to have access to Internet. Considering, the fact that inbound connections are desired traffic originated from Internet is recommended not to be restricted.

3.3 System Attributes

The system is required to fulfill some of the specification that will be discussed further, in order to be considered as a long term solution and not only a preliminary architecture to serve for short term purpose.

Reliability

Into an ideal scenario the system should only be unavailable when processes of maintenance are handled. Nonetheless short periods of unavailability are acceptable if the systems are capable of automatic restart or if the person designated with the maintenance is alerted when they are unavailable. Honeypots are not considered critical systems where the downtime comes with huge loss of data and adjacent problems, however some data is lost when the system is not available, but with proper handling this failure can be adapted.

Security

Running systems that are attacked by different individuals/groups is presenting a huge risk for the network and organization where honeypots are deployed. Therefore, the security is an important factor when design decisions are made. The honeypot network, is required to be isolated from the principal AAU's network and the outbound communication to be limited. Nonetheless, the system is desired to not be used as a jump station by the attackers to gain access of AAU's Services.

Portability

The system is not meant to be fixed to one machine, therefore a reasonable easy-changeable and portable solution is desired. Due to the fact that the hosting machine can suffer failures, is requested a solution where the honeypots can be implemented fast to a different location.

Maintainability

Maintainability is the property that if a failures occurs that system can be restored to working state into a time frame. Depending on the honeypots type that will be used the maintainability can vary from occasional to frequent, although the systems after deployment are not intended to be let unsupervised due to their high security risk. Is desired a system where the maintainability is not considered a main task after deployment, moreover maintainability being required only from time to time for changes or regular inspections.

On the grounds of the system requirements formulate in section 3.1 and consequently considering the system attributes the next chapter will offer an overview over the applicable hosting environments. Moreover, next chapter will also outline the chosen architecture for the system, by discussing network overview and design particularities.

[This page is intentionally left blank.]

CHAPTER 4

System design

The chapter is building on the system requirements formulated previously, but focus is directed to considerable solutions in order to meet the requirements. Therefore, the chapter from an organizational point of view is structured as the following:

- Overview of hosting architectures - to decide on a good hosting architecture, the existing possibilities are analyzed and described
- Overview of Docker - general knowledge on Docker is necessary for a better understanding of its capabilities and how can be applied in the project
- Overview of System Architecture - is detailing the chosen design together will all its the components
- Overview of Logs processing - is presenting a method to organize and visualize log files

4.1 Related Architectures

Considering that nowadays there are a lot of possibilities when it comes to systems' architecture, the section main purpose is to define the applicable solutions for honeypot deployments and to outline their strong and weak points. At the end of this section a conclusion will be traced and one type of architecture will be chosen for deployment.

4.1.1 Bare Metal Servers

As a bar metal server can be identified any physical computer that is used to run specific services without any interruptions for large periods of time. Additionally, bare metal servers are considered single tenant architectures, where the resources are not shared. In their architecture they are standalone device where access to hardware is permitted. Furthermore, one main advantage is that they provide isolation and performance being good hosting machines for systems where a lot of power is required. [16]

On the other hand, bare metal servers do not require layers of software to access them, they have direct access, unlike virtual environment where you need a least one additional layer of software. The layer of software in this case is the operating system (Windows or Linux). Nonetheless, physical server are not as scalable as a virtual environment, but they compensate by the fact that they are more powerful than a Virtual Private Server (VPS), another drawback for physical systems is the fact that they are a lot more expensive than a virtual environment. [16]

Consequently, physical servers can be seen as a good options for hosting honeypots, since they can offer direct access, performance and the isolation that is required. In this case any desktop computer can be used as a server since not a lot of power is required. However, decent hardware configuration can be desired for a good operation of the processes.

Moreover, a physical machine can offer the necessary isolation for hosting honeypots, since the honeypot will be easily isolated from the rest of the environment and any risks that can appear can be contained. For instance, a desktop system that has Linux as operating system can host multiple honeypots and with proper isolation can be considered a fairly secure testing environment. Nonetheless, if any error or damage is made the only affected system will the one where the honeypots are hosted.

4.1.2 Virtual Environment - Virtual Machines

Virtualization is used to describe the software that lays between the hardware and operating system. The abstraction level is hiding the physical resources from the operating system, and it is called a Virtual Machine Monitor (VMM) or a hypervisor. Since the hardware resources are controlled by VMM it is possible to run in parallel different Operating System (OS)es. For this case the hardware is divided between OSes into one or multiple logical units that are called Virtual Machine (VM), a graphical representation can be seen in fig. 4.1.

Virtual machines were developed due to the high amount of time and complexity that was required for maintainance task of large computers, but also that can run multiple processes in

parallel and this actually increase the efficiency of the environment. [30]

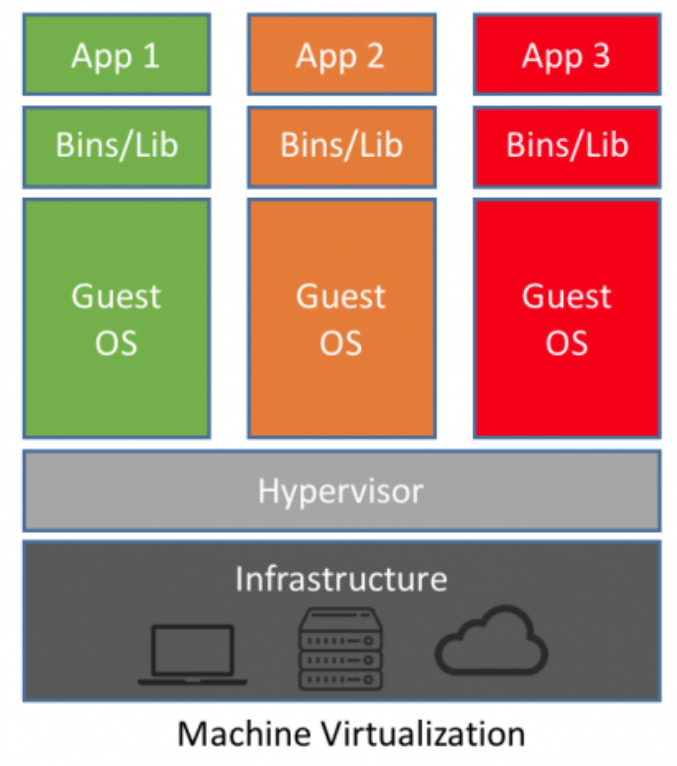


Figure 4.1: Virtual machine architecture presented in [5].

Some of the primary advantages that virtualization can offer are the resource sharing, in contrast to the bare metal systems discussed previously where all the resources are allocated to the process running on that machine, VMs share the resources such as memory, disk and network devices. Secondly, VM provide the layer of isolation that can be either come as an advantage by the fact that processes running on different VMs are not interfering with other process running on a secondary VM, or it can be seen as disadvantage that the processes cannot see programs running on other VMs. [30]

Furthermore, virtual environments have been seen as a choice for researchers in deploying honeypots since in this architecture they do not require additional physical systems. Nonetheless, using virtual honeypots is quite accessible to populate the network with different type of operating systems and services, and also it is easier to recreate or redeploy if they are compromised. [34]

However, there is one feature of the virtual environment that can lead to serious difficulties if is not managed carefully. Isolation can become a threat if the applications running inside one VM have access to the applications that run in other VMs. Additionally, by having a good isolation can prevent an attacker that have successfully gain control over one VM to extend his control to other VMs or to mitigate his attack to the underlying host resources. [30]

4.1.3 Virtual Environment - Containers

Containers were created as light operating systems inside the host operating system. They operate directly at the CPU without any requirement of a VMM, being limited in this way to the host Kernel for using the existing hardware. One main difference between VM and containers is that VMs share the resources between each other and every virtualization is using a different kernel specific for that operating system so every VM being a large consumer of the host CPU and memory. On the other hand, containers design the flexibility of the kernel is eliminated, by doing this the computation power is saved by using the operating system kernel. Moreover, a considerable difference comes also from the starting time, since a VM is required to boot up its private kernel container use the working kernel on the host machine [19]. In figure fig. 4.2 can be seen the architecture that is used for virtualization both for VM and for containers.

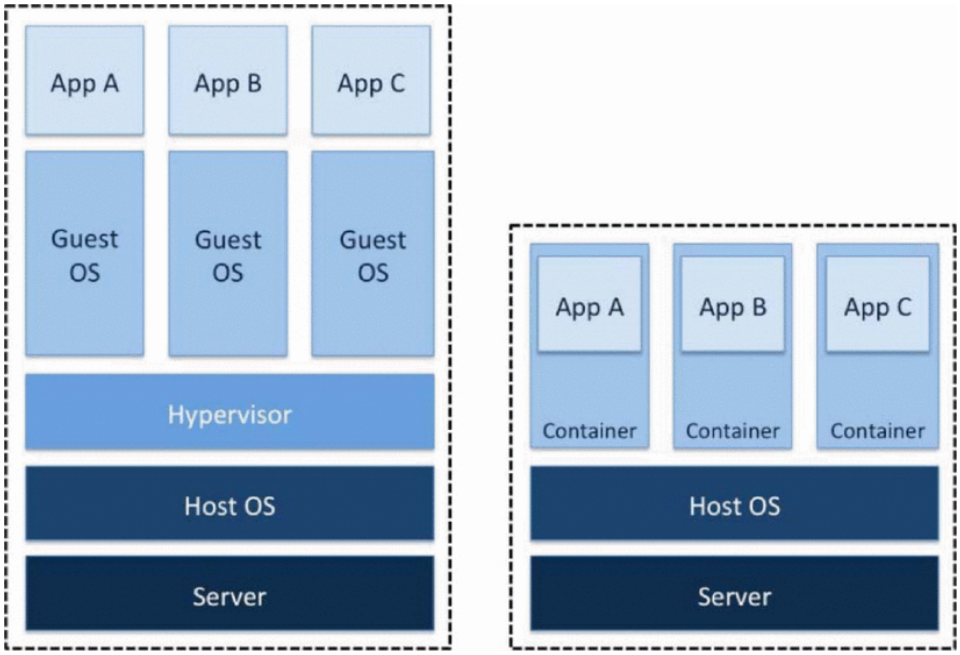


Figure 4.2: VM virtualization vs. containers method [20].

Containers architecture is not new, but their concept raise was due to DevOps industries and when Docker was introduced [27]. The main goal of developing into a containers is the possibility of sharing the necessary libraries that are required by a certain software product, additionally this feature adds a plus into the scientific area by facilitating the reproducibility of experiments[19]. There are different container technologies, therefore in the following paragraphs some of the most popular one will be named and described.

LXC - Linux Containers

LXC containers represent a light virtualization concept which does not need simulation of the physical hardware. LXC main advantage is that can run a complete duplicate of a Linux OS without the necessity of having an overhead hypervisor. Therefore, by sharing the same kernel with the host OS its file system and running processes are completely visible from the host. Nonetheless LXC containers use kernel namespaces to provide resource isolation and the container is seeing only the file systems and processes that were assigned to it. Moreover, CGroups

are used to handle resource facilities, for implementing processes and to offer network isolation POSIX capabilities are added. [4]

Docker

Docker comes as a continuation for the LXC container. Docker facilitates an open-source platform where everyone can create, upload or download the necessary containers and their full dependencies that can be found on the Docker Hub. Docker brings new features to manage data both kernel and application based ones. In addition, is adding more processes and advance isolation. The architecture is based on the server-client model, docker clients are communicating with the docker daemon which is working on the host OS. [19]

Singularity - High-performance computing (HPC) Container Platform

Singularity brings a new principle for containers, in terms of not requiring root privileges for using all the resources. Singularity is blocking the privileges escalation inside the container, and therefore if any script is needing root privileges inside container, root privileges are required also outside the container. Singularity is compatible with Docker, and has also its own implementation for Message Passing Interface (MPI) which permits to be use in HPC environments. [19]

On the grounds of earlier presented architectural concepts, the ideas that can be extracted are starting with the point that bare metal architecture will not be considered further than this point due to their lack of scalability, even if for the project scenario can be considered that the architecture offering the most isolation. Therefore, the alternative of a virtual environment is more suitable for deployment process, since is not presenting the necessity of new hardware and the architecture can be hosted on existing solution from the University. Additionally, when it comes to virtual environment containers are yet superior over VMs in terms of performance and scalability, and therefore they are considered a good fit for application deployment [18].

Overall, all containers technologies presented have similar capabilities, Docker will be preferred since is adding an additional layer of defense by its isolation. Moreover, Docker is providing process, devices and files restrictions and permits the docker container images to originate from local or remote storage. In addition, for the remote stored images they can be verified to avoid usage of any malicious ones. Nonetheless, Docker permits the user to tune their containers policies based on their needs. All the early mentioned feature come as run-time facilities. [22]
Since Docker environment will be used for the honeypots, the following section will be detailing more into docker technology features. Therefore, the next section will be focusing on docker containers, volumes and docker network properties in order to gain information that will be used during the design.

4.2 Docker

Docker is represented by a open source platform that is suitable for running, developing and distributing applications. Docker offers simple possibility to group all the necessary dependencies of an application into one package that can provide abstraction and that can

run despite of the hosting operating system. The packages used by Docker are called *containers*, and they run over the existing operating system, nonetheless containers exist from long time ago but Docker, successfully offered the isolation and abstraction that was investigated. [26]

Docker uses a client-server architecture, the client request to Docker's engine which is creating, building and running the containers for the application [10]. Docker client can be run on any platform, that as well goes for the Docker engine, in fig. 4.3 docker architecture can be seen:

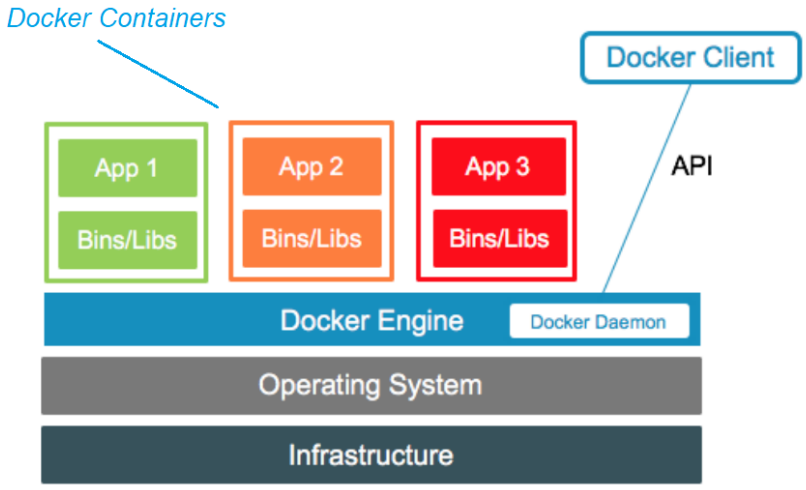


Figure 4.3: Overview of Docker architecture from [10].

4.2.1 Docker Containers

Docker is creating an instance of an image when a new container is created, the application code and configuration file are stored inside the container. Furthermore, Docker images are read-only and once created are not modified. Above the read-only images each container has a writable layer where runtime changes or privileges for files are stored and is called writable layer. Multiple isolated containers can run on the same host resulting in better usage of the hardware and also is minimizing the possibility of applications to interfere with each other. [7]

4.2.2 Docker Volumes

Docker Volumes are mechanism used to store persistent data generated by the containers. A *volume* can be created when a docker image is stared. Volume will be locally stored on the Docker host and that directory is mounted on the container as a *volume*. Volumes are entirely separated form the host core functionality, and they can be either assigned a random name by docker or can be named from the beginning. Moreover, if Docker is given the name of the volumes, is ensuring that they are unique inside one docker host. [9]

Nonetheless, if they are randomly named or custom made their functions are the same. Furthermore, volumes are preferred choice for persistent data because they are not increasing the size of container and data is still present outside, in fig. 4.4 a graphical representation of how volumes are associated with host file system.

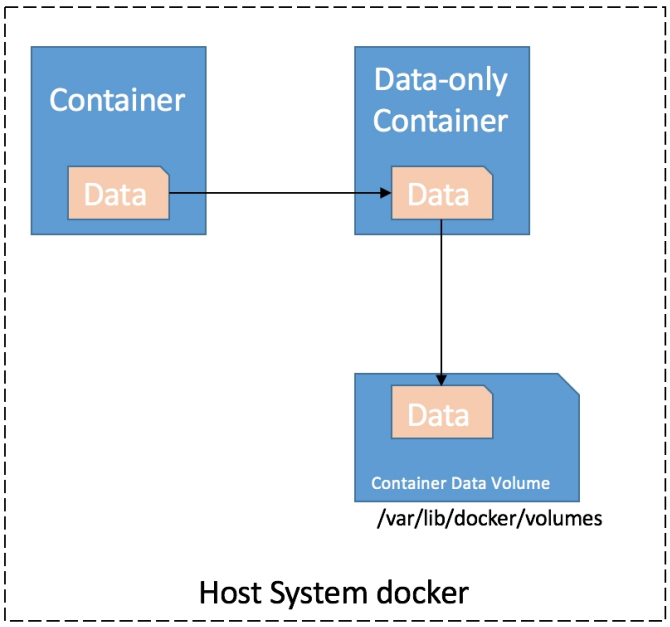


Figure 4.4: Overview of the honeypots integration with docker.

4.2.3 Docker Networks

Docker beside all the possibilities that is offering for the applications, it has multiple network infrastructure available for use. In the followings the available network drivers will be discussed:

Bridge - stands as a default network driver when no driver is specified. This type of network is used when applications run in standalone containers that can communicate [8]

Host - this driver removes the docker network isolation and the container is sharing the same network with the host [8]

Overlay - the driver connects different Docker daemons and enable services to communicate with each other. Nonetheless, *overlay* driver can be used for two standalone containers that are on different hosts to communicate, the driver reduces the necessity of using OS-level routing [8]

Macvlan - allows assignment of a MAC address to the container, make it appear as a device in the network. The traffic is routed by their MAC address. *Macvlan* is considered to be the best choices when the application is expected to be directly connected to the physical network [8]

None - this driver disables all networking. Most of the time it is used in relation with a custom network driver [8]

The following section is detailing into the proposed architecture and its integration with docker. Nevertheless, the used honeypots will be briefly named and attention will be distributed towards the network's architecture and how it is integrated with the AAU's network.s

4.3 System architecture

The system design is structured in regards with the requirements and constrains previously discussed in chapter 3, therefore will consist of the following parts: Hosting environment and Network features.

4.3.1 Hosting Environment

Based on the requirements discussed in section 3.1.2 and section 3.1.3 a VPS provided from the university will be used. The VPS will have Linux operating system installed as hosting environment, and afterwards the docker engine will be configured on the machine. Moreover, custom settings are necessary to provide isolation from the university network and this aspect will be further explained in section 4.3.3 and chapter 5.

4.3.2 Honeypots Integration with Docker

Docker will be used as a platform for deploying the honeypots, and each honeypot will be deployed inside one docker container. In section 4.2.1 were introduced docker containers and their advantages. Consequently, they seem to be a good fit for the architecture since by design they provide an extra layer of isolation.

Nonetheless, separation comes by default and every container will be independent from each other the architecture becomes more flexible. Considering, that an error can appear in one container, rest of them can still remain fully functional. In the followings, the honeypots that were chosen are discussed:

- **Cowrie** - exposing a SSH and telnet connection, with the possibility of logging and storing the IP-source, user-password combination and used commands;
- **Dionaea** - is capable of capturing malware samples and additionally emulates the protocols FTP, Trivial File Transfer Protocol (TFTP), Message Queuing Telemetry Transport (MQTT), MSSQL, MySQL, Session Initiation Protocol (SIP), SMB and Universal Plug and Play (UPnP). Dionaea is logging all the activity to JavaScript Object Notation (JSON) format log file;
- **Heralding** - design as a simple honeypot that is able to collect credentials. Heralding is emulating the protocols HTTP, Hypertext Transfer Protocol Secure (HTTPS), Post Office Protocol 3 (POP3), Post Office Protocol 3 Secure (POP3S) and Internet Message Access Protocol (IMAP);
- **Mailoney** - is a classical mail server that is exposing SMTP protocol with the possibility of storing the source-ip and commands used;
- **RDPY** - aiming to mimic Remote Desktop Protocol (RDP) from windows. It also has the possibility of logging the connection attempts and storing the source ip address;

The honeypots were chosen based on requirements from section 3.1.3 and section 3.1.4, and furthermore since all the above presented honeypots are LIHP the attacker interest is kept by

responding limited to the inserted commands. They are either mimicking a the real response or popping up a message that command is not available.

In relation with the requirements presented in section 3.1.5, section 3.1.6 and section 3.1.7 each honeypot is configured in a separated container, and the logs generated by the containers are saved on the hosting machine individually using docker volumes. In fig. 4.5 the honeypots integration with docker is displayed the approach is presented horizontally. Nonetheless, in section 4.3.3 the network structure will be detailed to offer an understanding about how the communication is working.

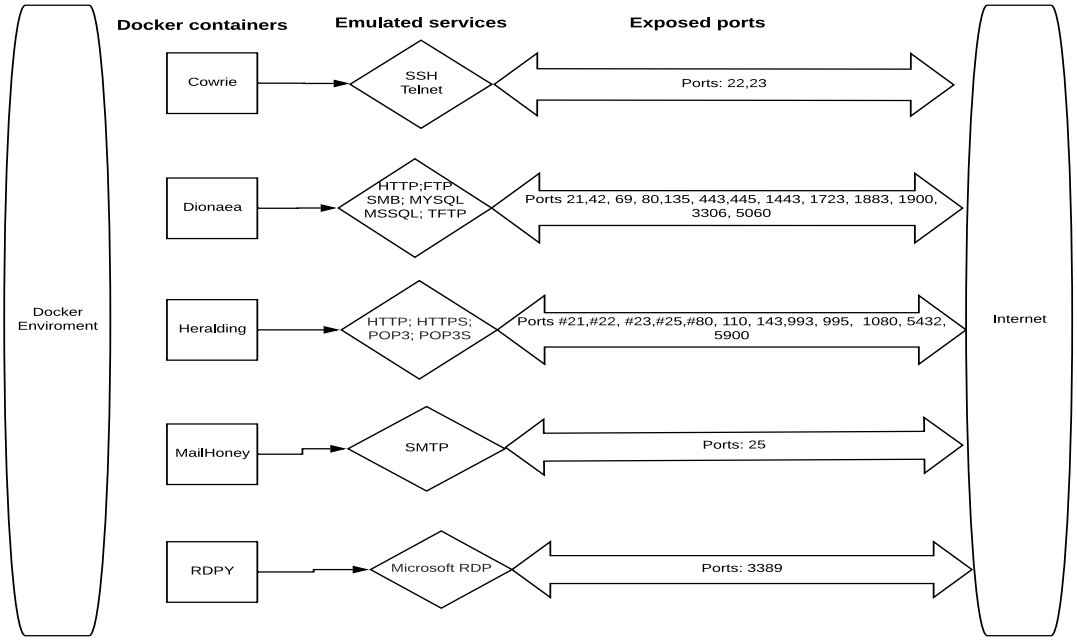


Figure 4.5: Overview of the honeypots integration with docker.

4.3.3 Network Design

Moving towards the network design as it was mentioned earlier, the VPS will be provided from the university. The VPS is configured on a small subnet(*130.225.239.113/29*) that resides outside AAU's firewall. Therefore, the server has a principal interface used for the SSH connection and other several secondary interfaces were created for the honeypots.

A network interface is represented by the connection that is form between a computer and private/public network. In most of the cases an interface is usually a Network Interface Card (NIC) without the requirement of having a physical form, since network interfaces can be implemented in software[24]. In fig. 4.6 an overview of the architecture is available:

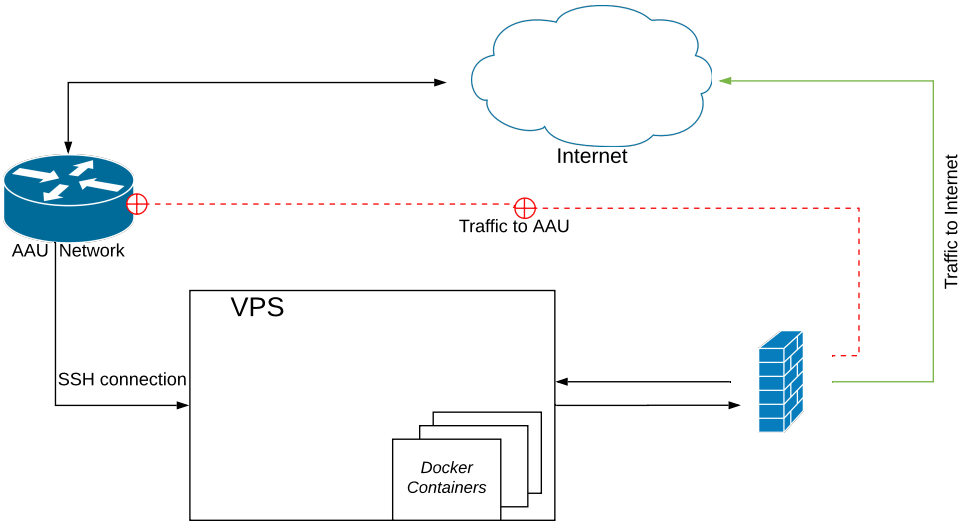


Figure 4.6: Overview of the honeypots integration with docker.

Therefore, secondary Ethernet interfaces were created for the honeypots *-130.225.239.115*, *130.225.239.116*, *130.225.239.117* and *130.225.239.118* and the honeypots will be distributed across them. By default the containers are connecting to the *Bridge* network that Docker is creating and the routing is done using *IPtables* which are detailed described in section 5.1.1. For this design this behavior was partially modified, for every container a static ip was allocated that is accessible from outside and also to integrate them with the AAU's network. Nonetheless, the same bridge default network is used, the only modification is made by assigning a static ip.

In theory, one IP address can be assigned to all the containers if the port mapping is done properly and no ports are allocated two times. However, for this specific project the full subnet was used in order to provide a better realism to the situation. It is hard to believe that a singular system is used to handle different types of protocols/services and all the ports from that services are open.

Since the interaction with the honeypots are stored into log files, a method to organize and analyze the files are presented in the following section. There are quite some options, but for this project the present architecture was used since it offers scalability.

4.4 Logs Processing

To fulfill also the requirement from section 3.1.8 a method to process all the logs files is required, furthermore all the honeypots are storing the activity in separated log files and therefore is desired to have a centralized processing method where all the logs are collected to facilitate the analysis. The system that is proposed in this report for collecting the logs and manage them is Graylog.

Graylog is an open source log management tool, that collects all the logs into a centralized location and makes them available, moreover it also offers a good performance and less downtimes [1]. Graylog has a backend functionality written in Java and user inteface written in Ruby-on-Rails, is able to collect *syslog* message over UDP and TCP, but it has also its personal protocol *GELF* that is simplifying the structuring of the log files.

For storing and passing the logs' messages, Graylog uses Elasticsearch where all the messages are indexed. Additionally, Graylog stores all the configuration in a MongoDB database [33]. Continuously, in fig. 4.7 Graylog architecture can be seen.

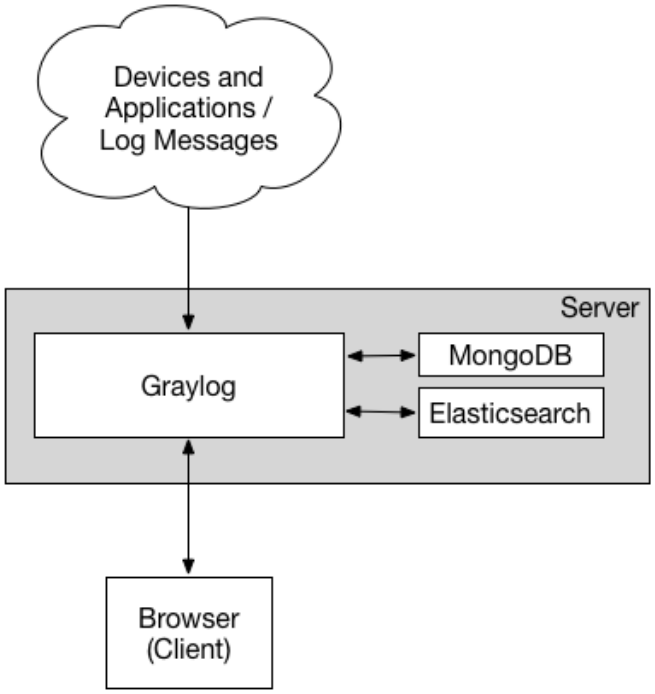


Figure 4.7: Graylog minimum architecture provided in [14]

Elasticsearch is a real time distributed tool mainly designed to organized data . To provide later easy access, it stores all the objects into a JSON format. All the documents that are stored are indexed by default, additionally it is a schema free storing systems so no fields for the data types needs to be defined prior adding data.[15]

The structure of elasticsearch is inspired from relational databases system with the difference that in elasticsearch the old fashion databases are replaced with indices. Moreover, indices are

collection of JSON documents in the same manner as SQL databases are collection of tables. Nonetheless, elasticsearch ensures to have a high availability of the data by keeping redundant copies of the it. When a new index is created is separated into one ore more shards that are kept on different node. Nonetheless, a node is considered any instance of Elasticsearch that is running. [15]

Docker containers will be hosting the honeypots and the data generated by them will be save into volumes for further analysis using Graylog. Main goal of honeypots is to lure attackers into connecting, and therefore additional attention needs to be put into the security policies that were implement to ensure that the design is separated from the university network and is not used as jump station. Next chapter will further detail the policies used and how they are set to ensure a better isolation.

CHAPTER 5

Docker Security

Honeypots apart from all the benefits that can bring when they are deployed inside an organization network, can come with several drawbacks. One setback is the fact that when a honeypot is configured into a network is an open door for attackers to that network and therefore honeypots needs to have proper security in place to not become more risk for the institution than an aid. Following chapter is describing the security measures that were used to ensure proper isolation from the AAU's network, the chapter is organized as follows:

- Overview of Docker Security - Docker architecture facilitates a secure design by isolation, grouping and by removing the necessity of additional softwares
- Overview of Firewall - in order to protect the university network and also to isolate the honeypot network Iptables were used
- Overview of Iptables settings - used to permit traffic to/from Internet and block anything originating from the VPS to AAU's systems

Docker increases the security by using containers that provides isolation. Layers are created between the applications, host and application, furthermore is protecting also the host by restring the access to it. Nonetheless, is a good practice to create resource restrictions around deployed applications, and docker enforces these practices by creating private file systems that permits separated user accounts.[10]

Continuously when a container is started docker creates a set of *namespaces* and *control groups* for that container. *Namespaces* refer to a feature of Linux kernel that the resources are partitioned while one processes sees one set of resources and another processes sees a different set of resources. Therefore, docker creates set of name spaces for the container to provide isolation from other running containers [10]. During the creation various namespaces are created:

- *PID Namespaces:* every program starts with a new and unique ID that is different from the host system.[10]
- *MNT Namespaces:* Mounted directory paths are uniquely created for each container[10]
- *NET Namespaces:* Every container has its view to the network stack, to avoid interference with other containers or socket access [10]

Consequently, when an attacker is connecting to honeypot that is hosted inside a container he can only find information about the processes, directories and network stack that are specific for that container without having any information of the host or other Linux containers that are running.[10]

Furthermore, Docker can control for each container the level of resources that is handling. *Control groups* permits docker to share the available hardware resources and to limit if necessary the use of them. For instance, the amount of memory that a container uses can be restricted so the host is not drained of memory, in case an attacker is trying to abuse the RAM memory. [10]

Linux systems offer the possibility of fragmentary user access, the root user is permitted access to all the feature, while non-root has more limited access with the possibility of growing his access with *sudo*. This may represent a security risk and Docker settings are configured to limit the Linux capabilities for cutting down the risk. The available list of commands for a docker container is less than in Linux and also the possibility of escalating the access to root is limited [10]. Moreover to support docker security architecture, custom users were set for the honeypots in the DockerFiles, and this can be seen in fig. 5.1.

```
1     addgroup -g 2000 cowrie && \
2     adduser -S -s /bin/ash -u 2000 -D -g 2000 cowrie && \
3
4     addgroup --gid 2000 dionaea && \
5     adduser --system --no-create-home --shell /bin/bash --uid 2000 --disabled-password
      ↪  --disabled-login --gid 2000 dionaea && \
6     chown -R dionaea:dionaea /opt/dionaea/var && \
7     rm -rf /opt/dionaea/etc/dionaea/*
8
```

Figure 5.1: Examples of the users created for the honeypots

Additionally, docker container run with limited set of abilities inside since all the jobs that require higher access are handle by the host system, nonetheless the normal container functionality is not affected. This default settings increases the container security by restricting access, and even if an intruder is capable of rising to root privileges is difficult to provoke host system damages [10]. Moreover, in this project LIHP were chosen and the response to attacker's command is limited, so no important damage can be made.

Finally, Docker implementation is considered to be an implementation where security is handled by default and where the need for other security application is not present. Therefore, the complexity is limited and possible system misconfiguration are avoided. As a principal conclusion Docker's environment is more secure for applications than running applications outside of containers, based on the level of protection and isolation a container is capable of offering. [10]

To further isolate the design and control the traffic flow towards university network, a firewall was implemented to limit the traffic originating from VPS to the university. Therefore, next section will present general aspects of firewalls and giving details about Linux Iptables that are used to imply the rules.

5.1 Firewall

Firewalls are instruments used by organization to impose security policies. Firewalls are capable of filtering network traffic in one of the layers of *ISO network model*. They are frequently used in the application, transport, network and data-link levels. Firewalls implement a set of rules known as *policies* that provide separation between networks, they also allow some traffic to reach its destination, while blocking the unwanted one. Nonetheless, firewalls can be good mechanisms for auditing and monitoring. They are offering a protection both to an outside attack but also to an inside one by preventing confidential information from leaking, or by preventing the users' access to unnecessary informations. [17]

The firewall used during this project is operating at the network layer of the ISO stack. Accordingly to honeypots' architecture is configured to accepted any inbound connections, but to block all traffic to AAU's network. Nonetheless, outbound traffic to Internet is allowed to minimize the attacker's suspicions that is being trapped inside a honeypot. In order to configure this policies, *Netfilter* from Linux kernel was used and further details about their management will be provided into the following subsection.

5.1.1 Netfilter - IPtables

Netfilter is a process that handles Linux network packet traffic, and *iptables* is the command used to apply configurations. For a simple understanding during this report the term that will be used to refer to this type of process will be *iptables*. Iptables operate by grouping traffic into chains, where rules are created that are afterwards save in tables depending on which of the chains are applied to. The rules are constructed based on patterns used to determine which of them is applicable for a specific group of packets and to assets what will be done with the packets that match a pattern [25]. For example in fig. 5.2 a command of how inserting a new rule can look like:

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80
  -j DNAT --to-destination 192.168.1.3:8080
```

Figure 5.2: Example of the command used to insert a new rule presented in[25].

Therefore, when a new rule is created, firstly the chain that fits into is specified, secondly the pattern and at the end what to happen with the packets if is not entering any rule. Iptables are targeting 5 types of packet traffic: *PREROUTING*, *INPUT*, *FORWARD*, *POSTROUTING* and *OUTPUT* [25]. In the following table these concepts of iptables will be explained together with their main features:

Chains	Packets' description
Input	Traffic that was delivered to a system
Output	Traffic that is generated from a system
Forward	Traffic that is going to a gateway computer and then is preparing to pass another one
Postrouting	Traffic that is preparing to leave the network interface
Prerounting	Traffic that have just arrive to the network interface

Table 5.1: Routing chains explained.

The chains are filtering traffic based on their origin and destination. Every chain is formed by tables where rules are constructed in order to match a pattern. In general a rule consists of one or more criteria that needs to assess how network traffic will be affected, if no pattern is specified all the traffic is considered. Targets are used to determine what is happening with the packets that are matched. There are 4 types of targets that are build into the iptables rules. In table 5.2, rules regarding the targets are explained:

Target	Description
ACCEPT	Permits packets to pass through the next stage of processing
DROP	Stop processing the packets completely. The packets are not checked for any rules, chains or tables. No information is sent back to the sender; REJECT can be used to provide some information.
QUEUE	Packets are send back in the userspace (not at the kernel level)

Table 5.2: Targets explanations used by Iptables.

Iptables rules that were used to isolated and filter traffic to/from AAU are detailed in next section, and also next section is providing a specific overview of the Ip ranges that are permitted to access the VPS's SSH connection.

5.1.2 Iptables - settings

Considering the idea that honeypots are designed to be attractive for attackers isolation and protection of the AAU's network is a mandatory. Firstly the rules for the *Input* chain will be

presented. The desired with this rules is to ensure that the interface - *130.225.239.114* used as a SSH connection is managed only by authorized people. Therefore, all unauthorized access is blocked without receiving any feedback for a better covertness. In the fig. 5.3 the rules applied can be seen. Moreover, is intended to allow only connection that originates from inside AAU's network or known IP addresses.

```
Chain INPUT (policy ACCEPT 11846 packets, 707K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      393 31575 ACCEPT     tcp  --  *      *       86.52.93.83          0.0.0.0/0           tcp dpt:1313
2     4762  294K ACCEPT     tcp  --  *      *       172.26.56.0/24       0.0.0.0/0           tcp dpt:1313
3        0     0 ACCEPT     tcp  --  *      *       172.26.50.0/23       0.0.0.0/0           tcp dpt:1313
4        0     0 ACCEPT     tcp  --  *      *       172.26.18.0/23       0.0.0.0/0           tcp dpt:1313
5        0     0 ACCEPT     tcp  --  *      *       172.26.82.0/23       0.0.0.0/0           tcp dpt:1313
6        0     0 ACCEPT     tcp  --  *      *       172.26.114.0/23      0.0.0.0/0           tcp dpt:1313
7        0     0 ACCEPT     tcp  --  *      *       172.26.8.0/22        0.0.0.0/0           tcp dpt:1313
8        0     0 ACCEPT     tcp  --  *      *       172.26.15.0/24       0.0.0.0/0           tcp dpt:1313
9        0     0 ACCEPT     tcp  --  *      *       172.26.72.0/22       0.0.0.0/0           tcp dpt:1313
10       0     0 ACCEPT     tcp  --  *      *       172.26.76.0/23       0.0.0.0/0           tcp dpt:1313
11       0     0 ACCEPT     tcp  --  *      *       172.26.12.0/23       0.0.0.0/0           tcp dpt:1313
12       0     0 ACCEPT     tcp  --  *      *       130.225.50.0/23      0.0.0.0/0           tcp dpt:1313
13       0     0 ACCEPT     tcp  --  *      *       192.38.55.0/24       0.0.0.0/0           tcp dpt:1313
14       1    40 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:1313
```

Figure 5.3: Rules used to restrict traffic for the SSH connection.

Secondly, no traffic originated from the VPS is allowed to proceed to the AAU's network, and therefore all the IPes used at the *Input* target are used again for the *Output* with one modification, for this chain the policy used is *Reject*. Nonetheless, all traffic to the Internet is allowed to not restrict the usage of the VPS. In fig. 5.4 the rules used format the chain are presented.

```
Chain OUTPUT (policy ACCEPT 91 packets, 9392 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.56.0/24      reject-with icmp-port-unreachable
2        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.56.0/24      reject-with icmp-port-unreachable
3        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.50.0/23      reject-with icmp-port-unreachable
4        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.18.0/23      reject-with icmp-port-unreachable
5        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.82.0/23      reject-with icmp-port-unreachable
6        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.114.0/23     reject-with icmp-port-unreachable
7        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.8.0/22       reject-with icmp-port-unreachable
8        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.15.0/24      reject-with icmp-port-unreachable
9        0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.72.0/22      reject-with icmp-port-unreachable
10       0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.76.0/23      reject-with icmp-port-unreachable
11       0     0 REJECT     all  --  *      *       0.0.0.0/0            172.26.12.0/23      reject-with icmp-port-unreachable
12       0     0 REJECT     all  --  *      *       0.0.0.0/0            130.225.50.0/23     reject-with icmp-port-unreachable
13       0     0 REJECT     all  --  *      *       0.0.0.0/0            192.38.55.0/24      reject-with icmp-port-unreachable
```

Figure 5.4: Rules used to restrict traffic to AAU's network.

The main chains that were used for firewall's rules were the Input and the Output chain. Continuously, rest of the chains are used by docker when the traffic is started. During the following chapter the actual deployment is detailed, together with the decisions made.

[This page is intentionally left blank.]

CHAPTER 6

Honeypots Deployment

The following chapter is detailing the honeypot setup, by presenting the implementation from two perspectives: an attacker perspective and a defender angle of the systems. In terms of structure the next chapter is organized as:

- Overview of Docker and IPTables settings - how docker is routing the traffic using iptables
- Attacker overview of the systems - an accurate representation of the honeypots' network and also how the systems are viewed when they are scanned
- Defender overview of the systems - security measures used in relation with the open source code
- Logs processing - integration of Graylog and elastic with the setup

6.1 Docker and IPTables

Apart from the files, processes and privileges containers offer separation and isolation also when it comes down to network structure. Each container is creating its own network interfaces, and therefore processes can use ports that are inside the containers. Docker's services are available by exposing the ports to outside world [3]. Docker is manipulating the network traffic by the used of the iptables. When Docker is started, the necessary iptables rules are created. Mainly, the Postrouting chain is affected by Docker.

Therefore, Docker is adding a new chain to iptables that were discussed, in section 5.1.1, to create a link between bridge network and the hosting environment. In figure fig. 6.1 the iptables docker rules can be seen.

```
Chain DOCKER (1 references)
num   pkts bytes target     prot opt in     out     source               destination
1       93  4836 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.2           tcp dpt:9000
2       89  4444 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.5           tcp dpt:3389
3        4   200 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.6           tcp dpt:25
4        6   304 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:27017
5        1    40 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:5061
6        1    44 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:5060
7        5   244 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:5432
8       80  3804 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:3306
9      240 11472 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.8           tcp dpt:2223
10       4   180 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:1080
11       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:1883
12    9569  573K ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.8           tcp dpt:2222
13      14   784 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:995
14       2    80 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:1723
15      32  1652 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:443
16     211 10132 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:1433
17       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:143
18       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:445
19       4   204 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:110
20       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:135
21     654 33840 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.4           tcp dpt:80
22      18   860 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:81
23       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:69
24       0     0 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:42
25       6   296 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:21
26       2    84 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.7           tcp dpt:20
```

Figure 6.1: Iptables docker chain.

Even though the default docker settings of the network are working, for this project scenario changes will be made since the ideas is to integrate the honeypot network with the existing AAU's network. In order to do so in figure fig. 6.2 can be seen the secondary interfaces that were created. The honeypots are designed inside docker containers, and therefore for every container is assigned a static IP address where different ports are exposed based on the emulated services. For a better mimicking of real network the honeypots are spread along all available addresses (*130.225.239.113/29*), since from an attacker perspective is hard to believe that one device has a lot of services/ports open.

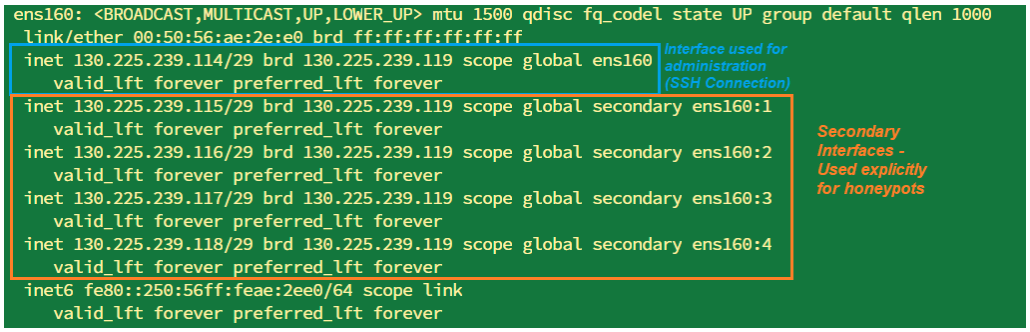


Figure 6.2: VPS network interfaces.

Once all Docker dependencies where installed on the VPS, next step is to install and configure *Docker-Compose* which is Docker's feature that can administrate multiple containers. Additionally, Compose file is a *YAML* configuration file where all the settings regarding the containers are declared ,and furthermore it offers the possibility of starting all the containers at once only by using a command. In fig. 6.3 and fig. 6.4 can be seen the docker-compose file that was created to support the easily transmissible architecture.

```
version: '2.3'

services:

# Cowrie service
cowrie:
    container_name: cowrie
    restart: always
    networks:
        ipv4_address: 130.225.239.115
    ports:
        - "22:2222"
        - "23:2223"
    image: "cowrie/cowrie"
    read_only: true
    volumes:
        - cowrie/downloads:/cowrie-git/var/lib/cowrie/downloads
        - cowrie/keys:/cowrie-git/var/lib/cowrie
        - cowrie/log/:/cowrie-git/var/log/cowrie
        - cowrie/tty:/cowrie-git/var/lib/cowrie/tty

# Dionaea service
dionaea:
    container_name: dionaea
    stdin_open: true
    tty: true
    restart: always
    networks:
        ipv4_address: 130.225.239.118
    ports:
        - "20:20"
        - "21:21"
        - "42:42"
        - "69:69/udp"
        - "81:81"
        - "135:135"
        - "443:443"
        - "445:445"
        - "1433:1433"
        - "1723:1723"
        - "1883:1883"
        - "3306:3306"
        - "5060:5060"
        - "5060:5060/udp"
        - "5061:5061"
        - "27017:27017"
    image: "dtagdevsec/dionaea:1903"
    read_only: true
    volumes:
        - dionaea/roots/ftp:/opt/dionaea/var/dionaea/roots/ftp
        - dionaea/roots/tftp:/opt/dionaea/var/dionaea/roots/tftp
        - dionaea/roots/www:/opt/dionaea/var/dionaea/roots/www
        - dionaea/roots/upnp:/opt/dionaea/var/dionaea/roots/upnp
        - dionaea:/opt/dionaea/var/dionaea
        - dionaea/binaries:/opt/dionaea/var/dionaea/binaries
        - dionaea/log:/opt/dionaea/var/log
        - dionaea/rtp:/opt/dionaea/var/dionaea/rtp

```

Figure 6.3: Docker-Compose file created for fast deployment of the honeypots arhictecture.

```
# Heralding service
heralding:
    container_name: heralding
    restart: always
    tmpfs:
    - /tmp/heralding:uid=2000,gid=2000
    networks:
    ipv4_address: 130.225.239.115
    ports:
    # - "21:21"
    # - "22:22"
    # - "23:23"
    # - "25:25"
    - "80:80"
    - "110:110"
    - "143:143"
    - "443:443"
    - "993:993"
    - "995:995"
    - "1080:1080"
    - "5432:5432"
    - "5900:5900"
    image: "dragas/heralding"
    read_only: true
    volumes:
    - heralding/:/heralding

# Mailoney service
mailoney:
    container_name: mailoney
    restart: always
    networks:
    - mailoney_local:
    ipv4_address: 130.225.239.116
    ports:
    - "587:25"
    image: "dtagdevsec/mailoney:1903"
    read_only: true
    volumes:
    - mailoney:/opt/mailoney/logs

# Rdpy service
rdpy:
    build: .
    container_name: rdpy
    restart: always
    networks:
    - rdpy_local:
    ipv4_address: 130.225.239.117
    ports:
    - "3389:3389"
    image: "amazedostrich/rdpy"
    read_only: true
    volumes:
    - rdpy/log:/var/log/rdpy

```

Figure 6.4: Docker-Compose file created for fast deployment of the honeypots arhictecture.

Consequently, fig. 6.3 and fig. 6.4 offer a good overview of the used honeypots. Moreover, it can be also seen the IP addresses that are allocated for every honeypot, exposed ports and the based image that was used for deployment. Due to the fact that the honeypots are storing information into the log files, the data is stored in persistent directories described into the compose file by *volumes*.

Continuously, next section will present the honeypots in the context of an attacker, where the actual network architecture can be seen and how it is integrated with the AAU, moreover the available open ports and the services that are running are visible.

6.2 From Attacker point of View

If an attacker is starting to scan AAU's network IP ranges the honeypot network will appear as small subnet with multiple open ports. The honey-network is visible since is configured outside the AAU's firewall to receive connections. For a better understanding of the honey-network, in fig. 6.5 can be seen how the network is connected with the main AAU's network that was previously presented in section 4.3.3.

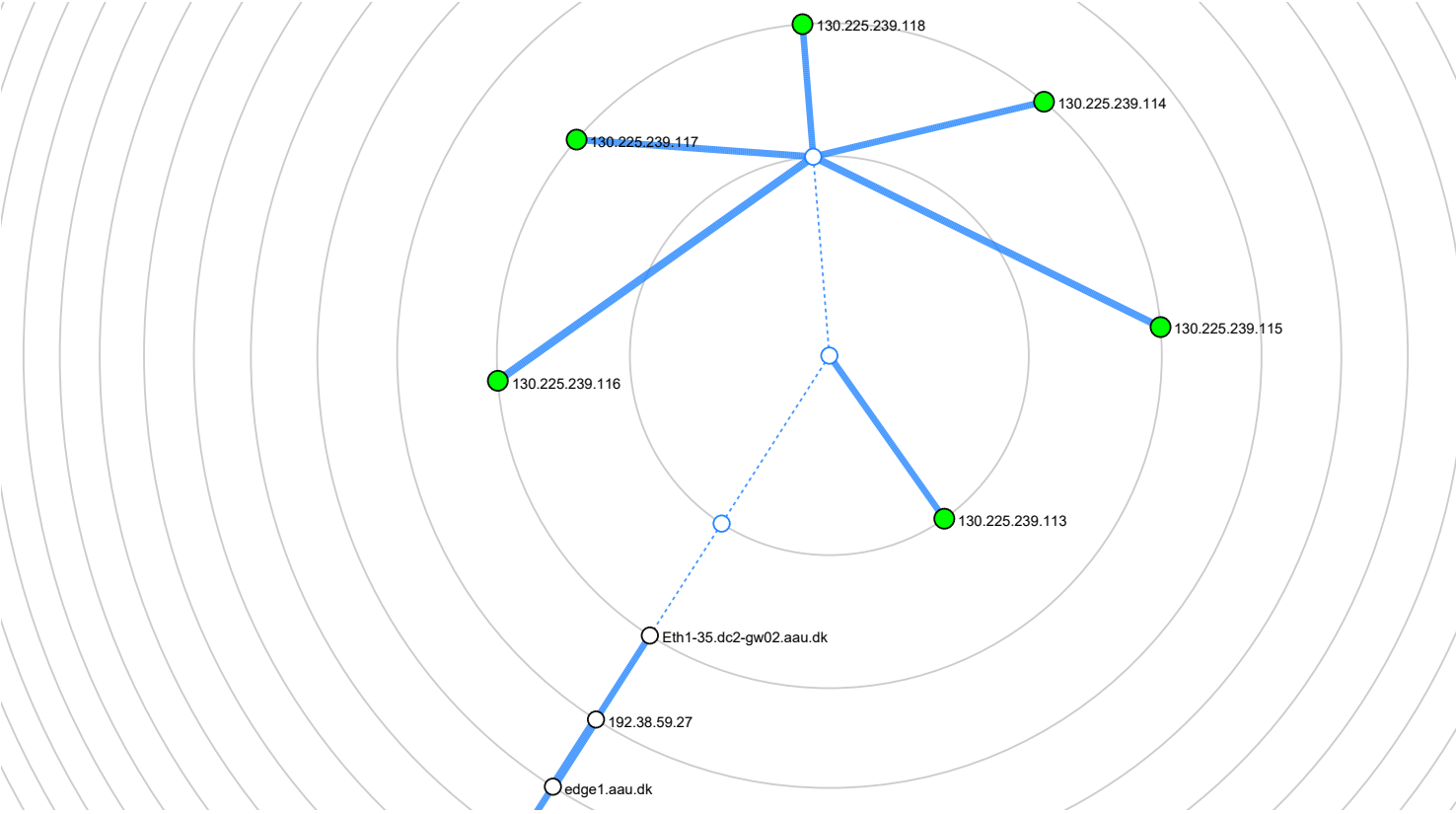


Figure 6.5: Honey-network from attacker perspective.

Furthermore, when an attacker is performing a scanning of the network the honeypots are seen as standalone systems that have different ports open based on the emulated services by the docker containers. If the scan is performed with *NMAP - Network Mapper*, which is an open source tool for network exploration and auditing [21], that is capable of various scan types the

level of details received is dependent on the scan method chosen. Therefore, if SYN scan is performed the results can be seen in fig. 6.6.

```
Nmap scan report for 130.225.239.115
Host is up (0.0027s latency).
Not shown: 990 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
23/tcp   open     telnet
25/tcp   filtered smtp
80/tcp   open     http
110/tcp  open     pop3
143/tcp  open     imap
443/tcp  open     https
995/tcp  open     pop3s
1080/tcp open     socks
5432/tcp open     postgresql

Nmap scan report for 130.225.239.116
Host is up (0.0024s latency).
Not shown: 998 closed ports
PORT    STATE    SERVICE
25/tcp  filtered smtp
587/tcp open     submission

Nmap scan report for 130.225.239.117
Host is up (0.0022s latency).
Not shown: 998 closed ports
PORT     STATE    SERVICE
25/tcp   filtered smtp
3389/tcp open     ms-wbt-server

Nmap scan report for 130.225.239.118
Host is up (0.0030s latency).
Not shown: 991 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
25/tcp   filtered smtp
42/tcp   open     nameserver
81/tcp   open     hosts2-ns
135/tcp  open     msrpc
445/tcp  open     microsoft-ds
1433/tcp open     ms-sql-s
1723/tcp open     pptp
3306/tcp open     mysql
```

Figure 6.6: Overview of the services and open ports based on NMAP SYN scn

Nonetheless this type of scan is considered to be rapid since is not opening a full TCP connection. A *SYN* packet is send, and if the port is answering back with a *SYN/ACK* packet is considered to be an *OPEN* port and the source is sending a *RST* packet. If the port is being *CLOSED* the destination host is sending back a *RST* packet, and nonetheless if the port is

FILTERED no feedback is received from destination.

Next section will further detail on the deployment by changing the viewpoint towards the defender perspective. Continuously are presented some of the aspects that were taken into consideration when the honeypots are deployed. It is important to notice the honeypots level of access and also the privileges.

6.3 From Defender point of view

Considering the aspect that containers deployed main purpose is to be attacked by malicious actors, containers' security must follow the best practices in order to avoid creating any opportunities to the attackers. Since not all the containers are used as honeypots, for instance *portainerio* which is hosting an User-Interface (UI) for containers administration was configured to run on port *9980* taking into consideration that is not in the most-common[21] ports that NMAP is scanning. This is done to avoid being detected on a regular scan.

In fig. 6.7 an overview of *Portainerio* is available. Nonetheless for all administration containers ports that are not in the most common NMAP are mapped and additionally iptables were configured to accept connections on these ports only from certain IP addressees.

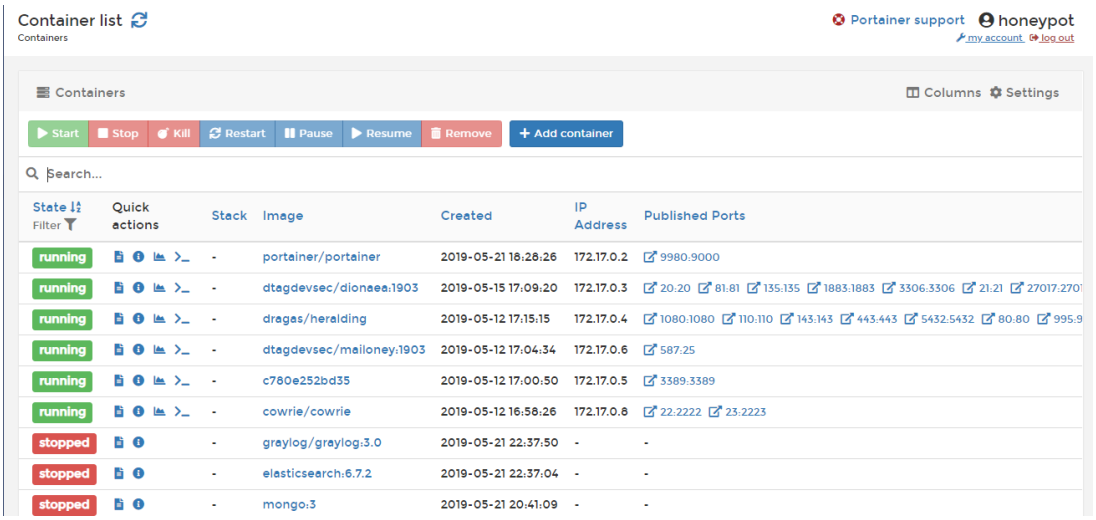


Figure 6.7: Portainer UI used to administrate the containers.

Honeypots were created from open-source images that were verified upon download to avoid Man-in-The-Middle types of attacks. The *DockerFiles* inside the image are created with special type of privileges, no *root* access is given to any of the created docker containers. For every image a new user and a group was created. The user is assigned without any base directory or root password to limit the attacker possibility of elevating to root privileges.

Once the honeypots start to attract attackers, the activity is logged into JSON files that are further utilize. Next section will discussed how the log files are organized to provide useful information about the activities in regards with honeypots.

6.4 Logs - Processing

As a result of container based architecture, the honeypot are set to store the activity in different log files and therefore a method to centralized the location of log files was developed. Firstly, a separated bridge network was created to facilitate the communication of the containers used for the analytics part and also to separated them from the honeypots.

Further, Graylog have been used as a log management tool, moreover after the logs are send to Graylog they will be saved in Elasticsearch to offer proper indexing of the data. In fig. 6.8 a graphical representation of the architecture used for processing the logs can be seen.

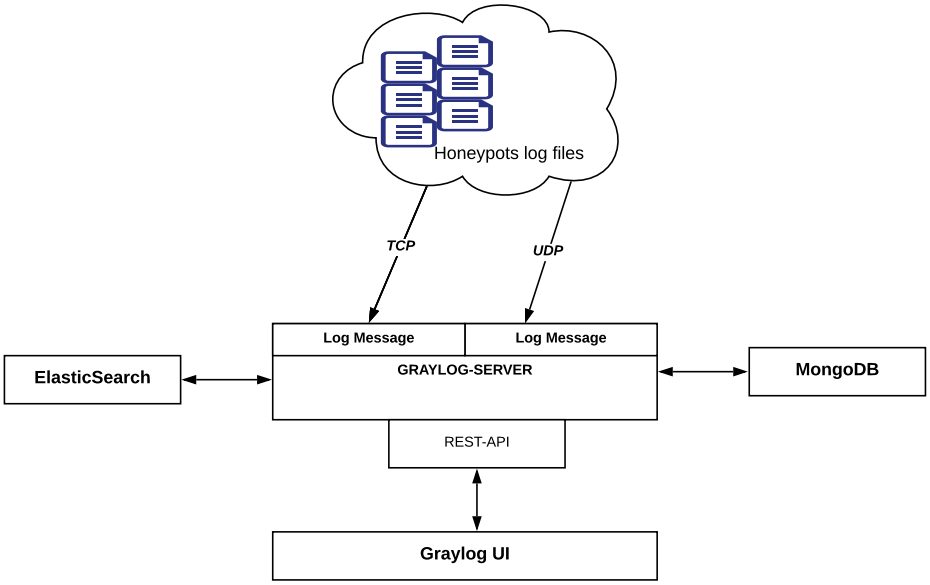


Figure 6.8: Overview of the log files parsing architecture.

Graylog is capable of receiving data of multiple formats and from different sources. For some of the honeypots the selected format was Graylog Extended Log Format (GELF), due to the fact that this is approximativelly similar with JSON format, but also due to the fact that part of the honeypots are logging the activity using JSONs. In fig. 6.9 an example of GELF format can be seen.

```
1 {"version": "1.1", "host": "example.org", "short_message": "A short message", "level": 5}
2
```

Figure 6.9: Example of GELF format.

In consideration of the fact that every honeypot are having different fields based on the honeypot type, the logs require to be normalized firstly to respect the GELF format and then to have as much as possible similar information to transmit. The transmission is happening over an open *UDP* port, netcat is used to transmit the data to graylog. In fig. 6.10 the script used to format the log files and to transmit them can be seen.

```
while read p;
do echo $p | nc -w0 -u 130.225.239.115 12201;
done <<< $(jq -rc
 ↪  '.timestamp = (.timestamp | sub("\\.[0-9]+Z$"; "Z")|fromdate|tonumber)' file_name)

```

Figure 6.10: Example of GELF format.

Additionally, log files can be sent as *RawText* and after the messages are received Graylog extractors can be used to organize the information, nonetheless structuring the messages into fields is important for Graylog analysis features. The data can be filtered based on regular expressions, Grok patterns, substrings or the message can also be devised into tokens.

Graylog is supporting flexibility by its various possibilities to inject data and its different supported formats, and moreover no limitations are enforced for the developers. Therefore, the following chapter will be presenting the activity that was registered by the honeypots using Graylog build-in statistics and graphs generators.

CHAPTER 7

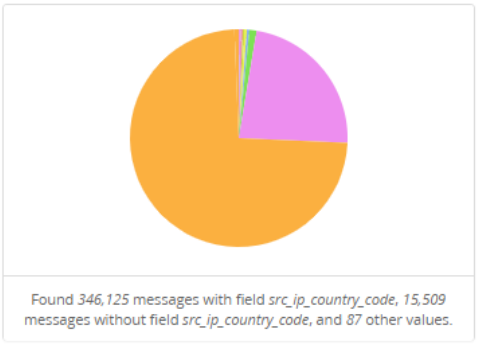
Results and Observations

The chapter will review the honeypots' outcomes by considering the total number of connections, the most attacked ports and also services that were having a high number of attempts. The chapter is organized in the following format:

- Overview of the honeypots - without being focused on a service or a specific honeypot
- Overview of Cowrie - activity tracked by the SSH honeypot is detailed
- Overview of Heralding - focused is put into the combination of user-password used and also on number of attempts by emulated protocols
- Overview of Dionaea - is detailing the attacked services, the relation between probed protocols and countries of origin for the attacks
- Overview of RDPY and Mailoney - is focusing on total number of attempts

The honeypots were started on 13th of May, and after 13 days there were registered a number of 361634 attempts. The majority of them were generated from SSH connections.

The majority of created sessions came from Ireland, nearly 74%, then the following 4 countries are Netherlands, Germany, Indonesia and Panama. Additionally in fig. 7.1 top 10 countries based on the number of attempts are displayed. The countries presented are considered to be the last hop of the attacks, since is not possible to identify from the logs if the attackers are using any proxies.



Found *346,125* messages with field *src_ip_country_code*, *15,509* messages without field *src_ip_country_code*, and *87* other values.

Value	%	Count
Top 10 values		
IE	73.70%	255,084
NL	23.05%	79,773
DE	1.07%	3,691
ID	0.36%	1,251
PA	0.35%	1,206
GB	0.29%	999
CN	0.27%	938
US	0.14%	481
BR	0.09%	327
KR	0.06%	223

Figure 7.1: Top 10 most active countries based on number of attempts.

Since the majority of the connections attempts came from Cowrie nearly 346346 of the total number, it is good to take a look at the protocol probed by not considering the SSH sessions since there is clear to be the largest ones. Therefore, an overview based on Heralding and Rdpy can be seen in fig. 7.2 there were a number of 10672 attempts distributed along the emulated protocols.

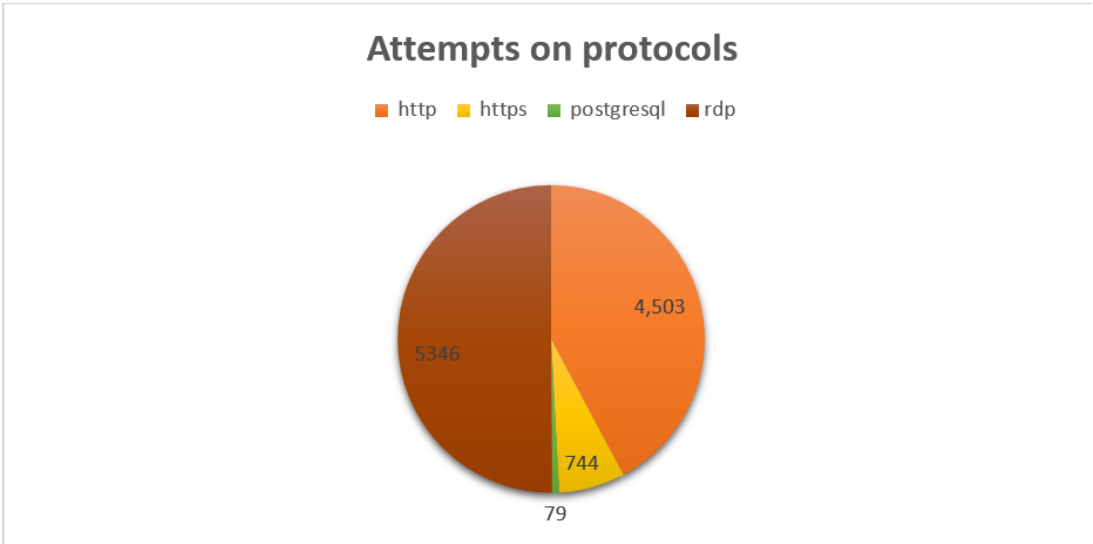
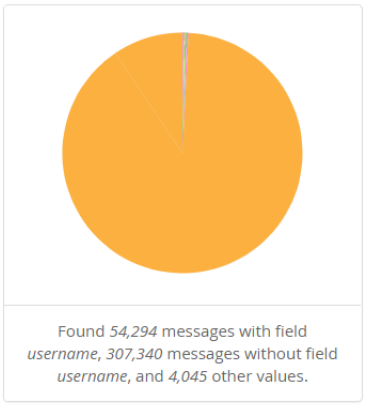


Figure 7.2: Attemps based on protocols.

Consequently, some of the protocols emulated require a username and a password, from the total number of attempts there were approximatively 54294 registered combination of username

and password. In fig. 7.3 the top 10 most used combination by their number of attempts can be seen.



Found *54,294* messages with field *username*, *307,340* messages without field *username*, and *4,045* other values.

Value	%	Count
Top 10 values		
root — admin	89.77%	48,742
root — 12345	0.25%	136
admin — admin	0.11%	59
user — user	0.09%	48
support — support	0.08%	45
admin — password	0.07%	38
admin — 12345	0.06%	34
admin — 1234	0.06%	33
admin — changeme	0.06%	31
admin — 123456	0.05%	27

Figure 7.3: Used combination for Username-password.

The following sections are further analyzing the results by going into details based on the honeypots and emulated services. The analysis is focusing on the combination of *user-password* that were used, together with the most probed services.

7.1 Cowrie

By far the most active honeypot in the presented time frame with 346, 346 registered attempts. There were found 96, 380 direct TCP connection requests to different targets and 48, 966 successfully connection were registered. Most of them are trying to connect to *ya.ru*, which is a Russian Internet company, on port *80*.The most used credentials combination were *root-admin* and most used password can be seen in table 7.1.

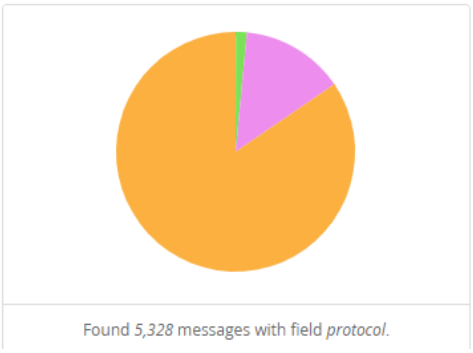
Password	Attempts
Admin	48713
12345	134
support	20
user	20

Table 7.1: Most used passwords by the number of attempts

7.2 Heralding

For this honeypot the most probed protocols are available in fig. 7.4, definitely *http* have gotten a lot of attention from attackers. Furthermore, when it comes to user-password combination, it was seen a tendency to use combinations of same user and password such as *admin-admin, root-root, postgresql-postgresql and Cisco-Cisco*. Nonetheless, in fig. 7.5 and fig. 7.6 are presented the most used usernames and most used passwords.

Values for *protocol*



Found *5,328* messages with field *protocol*.

Value	%	Count
Top 5 values		
http	84.52%	4,503
https	13.96%	744
postgresql	1.48%	79
socks5	0.02%	1

Figure 7.4: Heralding protocols overview.

Values for *username*

Value	%	Count
Top 10 values		
admin	77.25%	3,831
root	4.58%	227
super	2.26%	112
superadmin	1.79%	89
user	1.21%	60
fuck3g1	1.11%	55
support	1.11%	55
telecomadmin	0.63%	31
adsl	0.56%	28
cisco	0.56%	28

Figure 7.5: Top 10 most used usernames.

Values for *password*

Value	%	Count
Top 10 values		
cisco	2.07%	107
admin	1.84%	95
root	1.66%	86
Cisco	1.51%	78
support	1.08%	56
user	1.06%	55
mts	1.06%	55
airlive	1.06%	55
t3mp0Pa55	1.06%	55
Jkbvgbflf2014	1.04%	54

Figure 7.6: Top 10 most used passwords.

7.3 Dionaea

Dionaea is the honeypot that is emulating a high number of protocols. There were nearly 4619 connection and from the total number approximatively 3684 have successfully identify the attacked protocol. The distribution of attacked protocols can be seen in fig. 7.7, it can be seen that the most attacked service is *Microsoft SQL*.

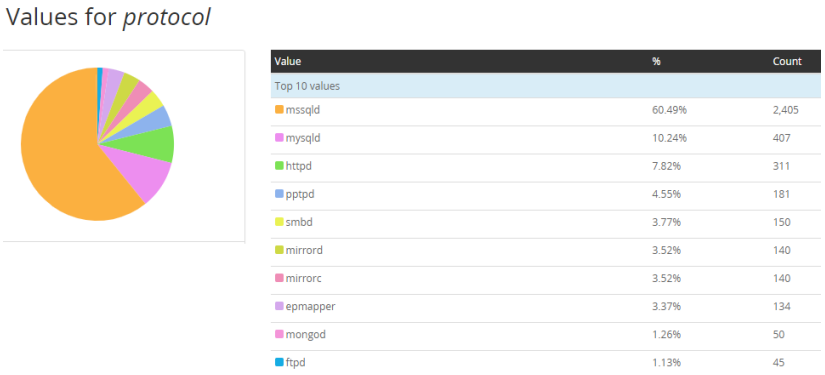


Figure 7.7: Top 10 most attacked services emulated by Dionaea

Moreover, the services early presented were also associated with the countries of origin. Ireland is still present at the top of the list, but is passed by China for this honeypot. In fig. 7.8 are presented the top 10 countries that have established connections with the honeypot.

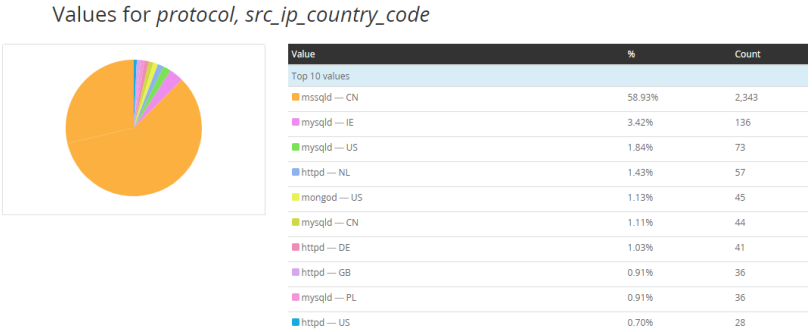


Figure 7.8: Top 10 most attacked services and countries of origin.

7.4 Rdpy and Mailoney

For the honeypot that is emulating and RDP protocol there were in the observed days a number of 5438 attempts. As for the Mailoney there were only 15 attempts, since the presented number is suspect low further analysis will be made to determine if the value was accurate or was generated due to an error made during the parsing of the logs.

Moreover the results need to be further investigated, and also is it worth keeping the honeypots active to observe if the general observations are changing once the time passes. However, it is expected a higher number of registered connections, but it is also interesting to see if the distribution of attacked services is kept or different protocols receive a higher number of attacks.

The following chapter is focusing into tracing a general overview for the project, underlying the principal observations.

CHAPTER 8

Conclusion

The project focusing into a method to integrate honeypots with an existing corporate network and it can be concluded now that there are a variety of choices. Additionally, the honeypots are acting as a decoy system, if a Danish Research Network IP ranges are scanned the honeypots are popping up as open services.

The approach demonstrated in this project was to integrate a honeypot network without having the necessity of connecting to AAU's network additional hardware device. Therefore, Docker was used as integration framework. Different types of honeypots were started inside Docker's containers providing robustness and isolation to the system. Nonetheless, Docker engine was installed on a VPS provided by the university.

Furthermore, to promote the attackers' desire to connect the honey-network was developed outside the university firewall. Consequently, the interaction with any systems from AAU is restricted but also the initiated connections are not filtered by the firewall. Additionally, honeypots outside the principal network act as decoy systems since attackers invest time and computational power while the real network is not attacked.

Placing the honeypots outside the university firewall have shown that can provide an excellent overview of the connections. However, one drawback that can be identified is the fact that no knowledge can be formed for what is happening inside the network, and therefore any on-going infection cannot be detected in this architecture.

Additionally, honeypots are excellent tools for gathering information about the attackers' origin country, probed systems and credentials tried. Therefore, the activity tracked by every honeypot is stored as persistent data by using Docker's volumes, further the individual log files generated by the honeypots are send to Graylog to be organized and analyzed.

It was seen for the services that require credentials a tendency to use the same combinations for username and password, and also generic username such as *root*, *admin*, *user* in combination

with passwords like *password123, 123456, pass123* needs to be avoided for the systems. Nonetheless it was also observed for the service that permitted connections that attackers are trying to use them as proxies. For the analyzed time frame the protocol that was the most attacked was SSH, but also in general protocols that are not secure such as HTTP were targeted.

Therefore, it can be concluded that honeypots can provide useful insights for an organization and they are worth deploying. Additionally, Docker containers are suitable architecture for hosting LIHP since the available commands are limited and the attackers are not having the necessary tools to escaped from the containers.

Moreover, LIHP are good statistical systems that can store the most important activity of the attackers, nonetheless utilizing LIHP is hard to distinguish between human or manual attacks, and moreover it is also complicated to extract attackers' TTPss.

To summaries, it can be stated that LIHP are valuable tools that are easy to maintain and deploy. Docker is facilitating the platform independence and also the virtualization of the system. Moreover LIHP can be considered systems that can bring valuable information for an organization without generating any high risk if the security features are setup accordingly.

List of Figures

List of Tables

[This page is intentionally left blank.]

Bibliography

[1] V. Agrawal, D. Kotia, K. Moshirian, and M. Kim. Log-based cloud monitoring system for openstack. In *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 276–281, March 2018. doi: 10.1109/BigDataService.2018.00049.

[2] Ateeq Ahmad, Muhammad Ali, and Jamshed Mustafa. Benefits of honeypots in education sector. *IJCSNS*, 11 (10):24, 2011.

[3] Charles Anderson. Docker [software engineering]. *IEEE Software*, 32(3):102–c3, 2015.

[4] D. Beserra, E. D. Moreno, P. T. Endo, J. Barreto, D. Sadok, and S. Fernandes. Performance analysis of lxc for hpc environments. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 358–363, July 2015. doi: 10.1109/CISIS.2015.53.

[5] Doug Chamberlain. Containers vs. virtual machines (vms): What's the difference? URL https://blog.netapp.com/blogs/containers-vs-vms/. [Online; accessed 31-May-2019].

[6] Cisco. 2018 annual cybersecurity report impacts on public-sector, 2018. URL https://www.cisco.com/c/dam/m/digital/elq-cmcglobal/OCA/Assets/Federal/2018-Annual-Cybersecurity-Report-Impacts-on-Public-Sector.pdf?ccid=cc000126&oid=rptsc008809&elqTrackId=64397bd4bdfd4bf6a6cde2dee70f3e6e&elqaid=4518&elqat=2. [Online; accessed 08-March-2019].

[7] Docker. What is a container?, . URL https://www.docker.com/resources/what-container. [Online; accessed 07-May-2019].

[8] Docker. Network drivers, . URL https://docs.docker.com/network/. [Online; accessed 11-May-2019].

[9] Docker. Use volumes, . URL https://docs.docker.com/storage/volumes/. [Online; accessed 07-May-2019].

[10] Docker. Introduction to container security - understanding the isolation properties of docker, August 2016. URL https://www.docker.com/sites/default/files/WP_IntrotoContainerSecurity_08.19.2016.pdf. [Online; accessed 19-April-2019].

[11] Telekom DTAG. Fruhwarnsystem, sicherheitstacho, 2013. URL http://www.sicherheitstacho.eu/. [Online; accessed 11-March-2019].

[12] F-Secure. Attack landscape h1 2018, 2018. URL http://images.secure.f-secure.com/Web/FSecure/%7Ba1352f14-be26-4fd1-bcc8-3c9bd6b20bd3%7D_Attack_Landscape-H1-2018.pdf. [Online; accessed 11-March-2019].

[13] Danish Defence Intelligence Service Centre for Cyber Security (CFCS). Foreign hackers threaten danish public research, 2017. URL https://fe-ddis.dk/cfcs/publikationer/Documents/TV%20forskning%20ENG.pdf. [Online; accessed 08-March-2019].

[14] Graylog. Architectural considerations. URL https://docs.graylog.org/en/3.0/pages/architecture.html. [Online; accessed 23-May-2019].

[15] S. Gupta and R. Rani. A comparative study of elasticsearch and couchdb document oriented databases. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 1, pages 1–4, Aug 2016. doi: 10.1109/INVENTIVE.2016.7823252.

[16] Inap. Bare metal vs. hypervisor: The evolution of dedicated servers. URL https://www.inap.com/blog/bare-metal-vs-hypervisor/. [Online; accessed 07-May-2019].

[17] Kenneth Ingham and Stephanie Forrest. A history and survey of network firewalls. *University of New Mexico, Tech. Rep*, 2002.

[18] A. M. Joy. Performance comparison between linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 342–346, March 2015. doi: 10.1109/ICACEA.2015.7164727.

[19] Á. Kovács. Comparison of different linux containers. In *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, pages 47–51, July 2017. doi: 10.1109/TSP.2017.8075934.

[20] Gregory M Kurtzer. Singularity 2.1. 2-linux application and environment containers for science, 2016. URL https://www.researchgate.net/publication/309129586_Singularity_212_-_Linux_application_and_environment_containers_for_science.

[21] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[22] A. R. Manu, J. K. Patel, S. Akhtar, V. K. Agrawal, and K. N. B. S. Murthy. Docker container security via heuristics-based multilateral security-conceptual and pragmatic study. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–14, March 2016. doi: 10.1109/ICCPCT.2016.7530217.

[23] Marcin Nawrocki, Matthias Wählisch, Thomas C Schmidt, Christian Keil, and Jochen Schönfelder. A survey on honeypot software and data analysis. *arXiv preprint arXiv:1608.06249*, 2016.

[24] Oracle. What is a network interface? URL https://docs.oracle.com/javase/tutorial/networking/nifs/definition.html. [Online; accessed 17-May-2019].

[25] Gregor N Purdy. *Linux iptables Pocket Reference: Firewalls, NAT & Accounting*. " O'Reilly Media, Inc.", 2004.

[26] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[27] Flávio Ramalho and Augusto Neto. Virtualization at the network edge: A performance comparison. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE, 2016.

[28] Atle Refsdal, Bjørnar Solhaug, and Ketil Stølen. *Cyber-risk Management*, pages 33–47. Springer International Publishing, Cham, 2015. ISBN 978-3-319-23570-7. doi: 10.1007/978-3-319-23570-7_5. URL https://doi.org/10.1007/978-3-319-23570-7_5.

[29] Atle Refsdal, Bjørnar Solhaug, and Ketil Stølen. Cyber-risk management. In *Cyber-Risk Management*, pages 33–47. Springer, 2015.

[30] Jyotiprakash Sahoo, Subasish Mohapatra, and Radha Lath. Virtualization: A survey on concepts, taxonomy and associated security issues. In *2010 Second International Conference on Computer and Network Technology*, pages 222–226. IEEE, 2010.

[31] ES-It Services. Network. URL https://it-wiki.es.aau.dk/wiki/Network. [Online; accessed 07-May-2019].

[32] UK Universities. Cyber security and universities; managing the risk. *Retrieved December*, 31, 2013. URL https://www.universitiesuk.ac.uk/policy-and-analysis/reports/Documents/2013/cyber-security-and-universities.pdf. [Online; accessed 03-March-2019].

[33] Risto Vaarandi and Paweł Niziński. Comparative analysis of open-source log management solutions for security monitoring and network forensics. In *Proceedings of the 2013 European conference on information warfare and security*, pages 278–287, 2013.

[34] Wira Zanoramy Ansiry Zakaria, S. R. Ahmad, and N. A. Aziz. Deploying virtual honeypots on virtual machine monitor. In *2008 International Symposium on Information Technology*, volume 4, pages 1–5, Aug 2008. doi: 10.1109/ITSIM.2008.4631930.

[35] N. F. Zulkurnain, A. F. Rebitanim, and N. A. Malik. Analysis of thug: A low-interaction client honeypot to identify malicious websites and malwares. In *2018 7th International Conference on Computer and Communication Engineering (ICCCE)*, pages 135–140, Sep. 2018. doi: 10.1109/ICCCE.2018.8539257.