National College of Ireland

Using honeypots to develop an understanding of intrusion techniques

MSc Research Project
MSc in Cyber Security
MSCCYBETOP

Darragh Goslin
Student ID: x20141416@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Darragh Goslin
Student ID:	x20141416@student.ncirl.ie
Programme:	MSc in Cyber Security **Year:** 2024
Module:	MSc Research Project
Supervisor:	Mark Monaghan
Submission Due Date:	Monday 12th August 2024
Project Title:	Using honeypots to develop an understanding of intrusion techniques.
Word Count: Page Count:	8369 31

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Darragh Goslin
Date:	10/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

Using honeypots to develop an understanding of intrusion techniques

Darragh Goslin

x20141416

Abstract

With modern systems relying upon cloud infrastructure to support enterprise networks it essential to protect these systems from attacks. As cyber threats increase in complexity it is necessary to develop methods to detect how a malicious threat actor will breach a network. Through building a better understanding of the threats we can make efforts to harden and protect the network. A honeypot is used to achieve this goal. A honeypot is a rogue endpoint on a network. It is an endpoint that is intentionally left vulnerable. By allowing this rogue system to be breached we can monitor and see the methods a malicious actor might employ. We can also see how a malicious actor will behave once they gain access to a vulnerable endpoint.

Keywords:

Honeypot, Cloud Computing, Intrusion Detection

1 Introduction

Threat intelligence is essential for improving a security posture. Threat intelligence is gathered through a number of methods. This can include government agencies, private sector organisations, and open-source information. One source for gathering threat intelligence is the use of honeypots. The honeypot will be left vulnerable to allow for threat actors to scan and attempt to gain entrance. The methods used can be analysed and studied to build a better understanding of what vulnerabilities are available.

A successful cyber threat will ideally go unnoticed as they gain access. Unless the goal is distribution for issuing ransomware where the goal is to make the threat actors action visible to the victims. A honeypot can be modified to collect and gather data on a malicious actors actions in specific circumstances. Honeys can be classified into different categories such as low interaction to high interaction which will provide threat actors varying degrees on interaction. The implementation of honeypot will be created in a manner that will not put any production devices at risk. For the purposes of research a honeypot can be put in a cloud environment isolated from a researchers devices. The data gathered can be used to create a better picture of how threat actors will interact with vulnerable systems. This can include information such as which ports are used, what IP addresses the threat actors originate from, what passwords are compromised and what countries or times the attacks take place. With a better understanding of the attacks the security posture can be improved by closing ports, restricting geo location access and blocking malicious IP addresses. The file paths or files accessed by a threat actor can identify vulnerable areas for a security team to review. To

gather threat intelligence a number of honeypots will be deployed in the cloud. The honeypots will have different configurations to test the different approaches by threat actors. The attacks are often implemented by bots which can lead to repetitive actions. These actions can be recorded in logs and the frequency of attacks measure.

2 Related Work

Honeypots are a suitable method for collecting research data. Bhagat and Arora (Arora, 2018) noted that using honeypots to gather small amounts of data makes it easier to analyse and understand. By having a honeypot with a limited scope it gives a more concise area to focus upon. However it is important to note that the honeypot is not to be seen as a way to solve any issues or vulnerabilities on a network. Any interactions with the honeypot should be considered malicious. A honeypot involves silent detection by tracking malicious actors Ips and actions which will later be reviewed and analysed. A core principle in a functioning honeypot is the ability to avoid detection.

While they cannot be used to solve vulnerability issues Negi, Garg and Lal (P. S. Negi, 2020) did argue that they can be utilised to divert malicious threat actors away from critical frameworks to the honeynet framework. They did agree that a key concept of a working honeypot involves silent detection. A low interaction honeypot will be easier to detect no matter how good the emulation is. Eventually a skilled threat actor will discover their presence. As they are often based upon open-source frameworks a skilled threat actor will have access to the same resources and learn to spot the signs of a honeypot. Some restraints will need to be taken when implementing a honeypot which may include setting up the decoy system in an isolated network. This kind of deployment means they are suited to cloud infrastructure for detecting intrusions. The authors went further suggesting any cloud framework should have a honeypot deployed.

The deployment of a honeypot needs consideration. This was further expanded on in Architecture of the Honeypot System for Studying Targeted Attacks (Lapin, 2018) where the authors identified what information the administrator is seeking to gather. The design of a honeynet will differ from typical system architecture as they are designed in a way to allow for the study of malicious actors attacks.

The information we are seeking to achieve include:

- analyzing step-by-step intruder’s actions within the information system
- identifying targets pursued by intruders and potential targets of encroachment.
- detecting previously unknown malicious software used by intruders
- collecting various information about the intruder’s infrastructure (server, address, etc.

Other factors that need to be considered are that every endpoint needs logging capabilities with self-storage available. The logs collected should be protected from unauthorised access and preserved. The true function of the honeypot should be hidden from detection

After the malicious actor connects to the honey trap, Polyakov and S. A. Lapin groups the actions into:

• single use instance of the Honeypot system.

• using the Honeypot system as a gateway for further illegal actions against other information systems.

• usage of the Honeypot system instance to commit unlawful actions in relation to the workstation's LAN of the enterprise.

The threat actor needs to be able to gain access to the system without arousing their suspicion. This could involve providing fake file directories or changing the default hostnames and user names.

Each instance of the Honeypot system must include tools that will allow it to:

• monitor network activity. Their main task is to obtain a complete copy of the transmitted data, as well as to collect information on emerging network connections, such as IP addresses, port numbers, domain names, etc.

• monitor the events of the file system.

• track the activity of emerging and existing processes in the system, as an intruder can use malicious software which could generate new processes in the operating system.

In collecting the data it is essential for our honeypots to work efficiently and it is important that they maintain their activity around the clock. (Chandane, 2021) argue that in the design process a decision needs to be made towards whether the approach should be a qualitive or quantitative model. A honeypot can be designed so that it would difficult to gain entry and have many decoy services to entice an attacker. However, it would have limited results whereas a quantitative model will entice more treat actors. By having a larger amount of data we can build a stronger model on threat actor behaviour.

It is necessary to take steps to address this issue as the honeypot could result in misrepresentation, ineffectiveness and inadvertently allow detection by an attacker. An ineffective honeypot is a challenge for researchers. A researcher must be able to work with limited resources in the most efficient manner. A passive honey can be more efficient that an active honeypot where incidents will not require intervention by the researcher. To assist with maintaining around the clock access and that resources do not fall short a cloud environment will be preferential to an on-premises infrastructure.

In Long-Term Study of Honeypots in a Public Cloud (al, 2022) the authors states that a public cloud environment are convenient for computation and storage resources. However they can also be beneficial for malicious web based attacks which results in cloud based virtual machines often being attacked. This study showed that low interaction honeypots were repeatedly attacked when compared to medium interaction honeypots. Attackers seek to compromise cloud resources to launch their attacks or run command and control operations. A cloud resource could be used for mining cryptocurrencies. An analysis of threat behaviour actions may be divided by looking at the traffic interacting with the honeypot and then secondly by reviewing the content of files dropped onto honeypots.

More time spent on the system and interacting with the honeypot allows for more data to be collected. The data gathered from honeypots can lead to a type of feedback loop where the information that is collected is then used to further improve the interaction, deception and camouflage of these systems. Which is a point raised by Ali (Ali, 2022).

They believed there are two different areas for approaching the honeypot: attack versus secure. In the first scenario a fake endpoint can facilitate man in the middle attacks by a malicious actor. In the second scenario the honey is used for tracking the malicious actors and the data gathered used for securing a network. With increasing geopolitical crises, cyber attacks will continue to rise in frequency and sophistication the benefits of using honeypots to secure a network will persist. However this has some difficulties. They only give an overview of the malicious actors activities against the network. So if the threat actor was to bypass the honeypot when penetrating the network, then their actions will be oblivious.

By passing the honeypot is not the only concern, the other is that the malicious actors may become aware of the honeypot. Some indicators that may give away its presence is using ‘Fingerprinting’. An attacker will expect a web server to respond with typical errors and returning common HTML syntax but if the honeypot responds with incorrect HTML commands or has spelling errors it will give away the deception. By setting up a honeypot in the IT network a degree of risk is being introduced to the environment. A simpler honeypot will have a lower risk. The risk can be reduced or mitigated by how the honeypot is built and deployed.

Dowling, Schukat and Melvin (S. Dowling, 2017) agreed that the operation of honeypots can have ethical and legal concerns. A compromised honeypot could be involved in further attacks. The majority of attacks on honeypots are caused by botnets. These provide a method to threat actors for global cyber attacks. The attacks take the form of infection and control. A botnet is a large network of compromised machines which are under the control of a single command and control (C & C). The machines are common everyday machines that are inadvertently participating in the attacks. It is difficult to take down the botmaster as the C & C connection is changed.

To combat the bots we need to understand the methods for attacking**.** The data gathered can be used for calculating the probability of attacks occurring at certain times and the different types. (Mittal, 2024) explained that a botnet is a group of computers that is controlled and monitored by malicious actors. Hackers use bots to send malicious code. The botnet will infect a device that is connected to the network. Once the victim device establishes a connection to an existing command and control server the victim device can be used to spread malicious code under the control of the command and control server. The code

is often injected using protocols like HTTP, P2P and other remote tools. A common attack is brute force method for cracking passwords. It is a hit and error way of guessing the victim device password.

The threat actors will be seeking Zero day attacks which are undiscovered vulnerabilities that have yet to be patched. (M. Başer, 2021) noted they are a challenge for traditional cybersecurity methods where they invalidate any of the actions taken. Honeypots can be used to discover the attacks. The deception machine can be divided up based upon the classification. Baser stated their two divisions of classification:

i) Classification based upon installation which is whether the honeypot is production or research.
ii) Classification based upon interaction. This depends on the level of interaction by the malicious actor. Başer states there are three types which are low, medium and high interaction.

A high interaction honeypot will replicate many services and provide a sphere of activity for the threat actor. Whereas a low interaction honeypot will provide the least amount of interaction. It may just replicate limited number of services such as SSH remote access. Teknet and SSH are common protocols used for attempting unauthorised access to machines. Threat actors will frequently use default credentials to attack secure systems. A low interaction honeypot that utilises Telnet and SSH connections will allow for gathering information on how a malicious actor interacts with a endpoint via those connections.

In Analyzing the Attack Pattern of Brute Force Attack on SSH Port (A. Subhan, 2023) the authors agreed that in identifying the attack patterns one of the most frequent targets are remote access services such as Secure Shell (SSH). SSH is common used in Linux and Unix-based systems as an access shell. SSH is a network protocol used for exchanging data through secure channels. SSH operates at the application layer of the TCP/IP protocol. It uses the TCP port number 22. Threat actors will often target SSH ports using brute force. The attack rate can reach up to 25% in Denial of Service (DoS) attacks. The impact of the brute force attack is influenced by the complexity and length of passwords. The time required to impact them can vary from seconds to years.

(Dadarlat, 2018) agreed that a honeypot can discover correlations between successful attack methods and simple login credentials used during ssh connections. Analysis of honeypots over a longer period of time will show frequency of attacks. Long period analysis can be used to measure the effectiveness of security policies. For example a complex password policy can be tested by running a honeypot that prevents simple passwords.

A challenges to be addressed with running the research honeypots will be the volume of data which (S. Sunil, 2023) agreed by stating noted that as log become larger manual log analysis becomes more challenging. Concerns were also raised over the unstructured nature of logs and how they are often not in uniform design and that there is no set way for deciphering log files. A common concern raised is that they logs could be injected with malicious code or scripts in an attempt to corrupt the logs. Proper parsing will be needed to protect the logs. (Georgieva, 2020) agreed that information security is an unsolved challenge for organisations managing the volume of security logs. Georgieva proposed beginning by using filtering using relevant features for selection. The elimination of irrelevant features can

speed up the process. Content and context analysis is necessary with so many attack vectors available. Different models being built up using a variety of feature filters will provide more information on attack vectors.

3 Research Method

3.1 Implementation

To collect research data honeypots will be deployed on a cloud environment. The honeypots installed on vulnerable virtual machines with the capacity to log connections to SSH and Telnet on ports 2222 and 2223. Any actions made by the threat actor will also be logged.

The research can be broken down into these phases:

- System Assembly
- Data Gathering
- Data Processing
- Data Analysis
-

3.2 System Assembly

This step will involve creating the environment for the research and configuring the honeypots with different settings.

Hosting Environment

A Cloud environment will be used as it is better suited for hosting the honeypots and will provide the researcher a degree of protection in the event of a breach. Should a threat actor escape the honeypot or if the machine is compromised it will be easier to shut down and there will be no risk of actual data being at risk. Virtual machines allow for quick deployment. The cloud providers have images of the virtual machines that can be selected and configured to suit the requirements of the honeypots.

3.3 Data Gathering

The honeypots are left running to collect and monitor activity by threat actors. WinSCP will be used to gather the data files. Once a connection is made to the honeypots the log files will be downloaded to the research device. To reduce risk when collecting any suspicious files that have been uploaded to the vulnerable machines, this will be performed from a virtual machine. Windows Sandbox will add an extra layer of protection so that the files can be scanned and confirmed to be safe for interaction.

Cowrie is a medium interaction honeypot. It will simulate that ports 2222 and 2223 are open and allow SSH and Telnet connections. When a successful connection is made the threat actor will in a simulated Linux directory.

3.4 Data Processing

The data processing phase will involve transforming the logs to a form that will allow for analysis. Python scripts will be used to break down the log files and extract the relevant data for analysis. The connections will be extracted from the logs files and exported to a file for analysis. The passwords used by the treat actors will also be extracted and stored in a file. The IP addresses will also be sought from the log files. Suspicious files will be uploaded to VirusTotal to confirm whether they are known threats and to retrieve the Sha256 number.

3.5 Data Analysis

The resulting data from the processing is reviewed to gain any key findings or insights. Excell and Python will be used to perform the analysis of the data.

3.6 Tools

Cowrie

Ubuntu

Python

Excell

Winscp

Putty

3.7 Evaluation

At the end of the process there should be a collection of malicious Ips. To further confirm the Ips are threats they will be checked against existing collections of malicious Ips such as abuseipdb and virustotal. The research should provide logs of data that include not just Ips but the credentials and passwords that have been breached.

The research will be a greater success if the actions and the commands of a threat actor are recorded in the logs. We will then be able to better understand the methods taken by the threat actors. To attain different forms of data one of the honeypots will be configured so that the most common passwords will be blocked

3.8 Ethical Issues

The honeypots need to be implemented in a manner that will prevent other systems from being breached. There are also concerns that should the systems become compromised they could be used to host malicious attacks or as a launching ground for malicious activity.

Failure to identify these risks and threats to a network could lead to the honeypot becoming it compromised and the researchers device.

4 Design Specification

4.1 Cloud Provider

The underlying elements in the design will be to isolate the honeytraps from other systems and to leave them vulnerable to entice an attacker. The honeypots will be hosted on a cloud provider to keep them isolated from the researchers devices. Should an attacker escape the device they will not be able to retrieve any actual data or launch attacks onto the researchers devices. If the machine is compromised it will be easier to shut down.

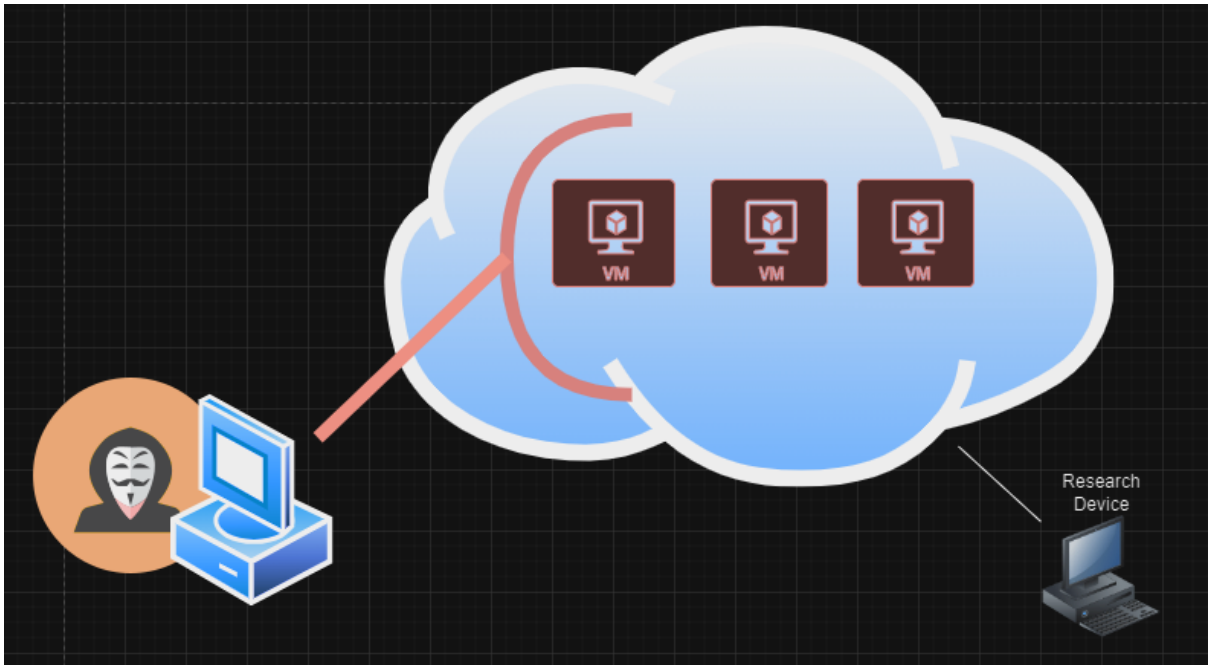


Figure 1: Three vms in cloud

On the Cloud Provider the honeypots will be implemented onto three separate virtual machines. Each vm will be hosted on a separate projects space to further isolate the devices. The honeypots will be installed on virtual machines with Ubuntu installed. This allows for quick installation and easy configuration.

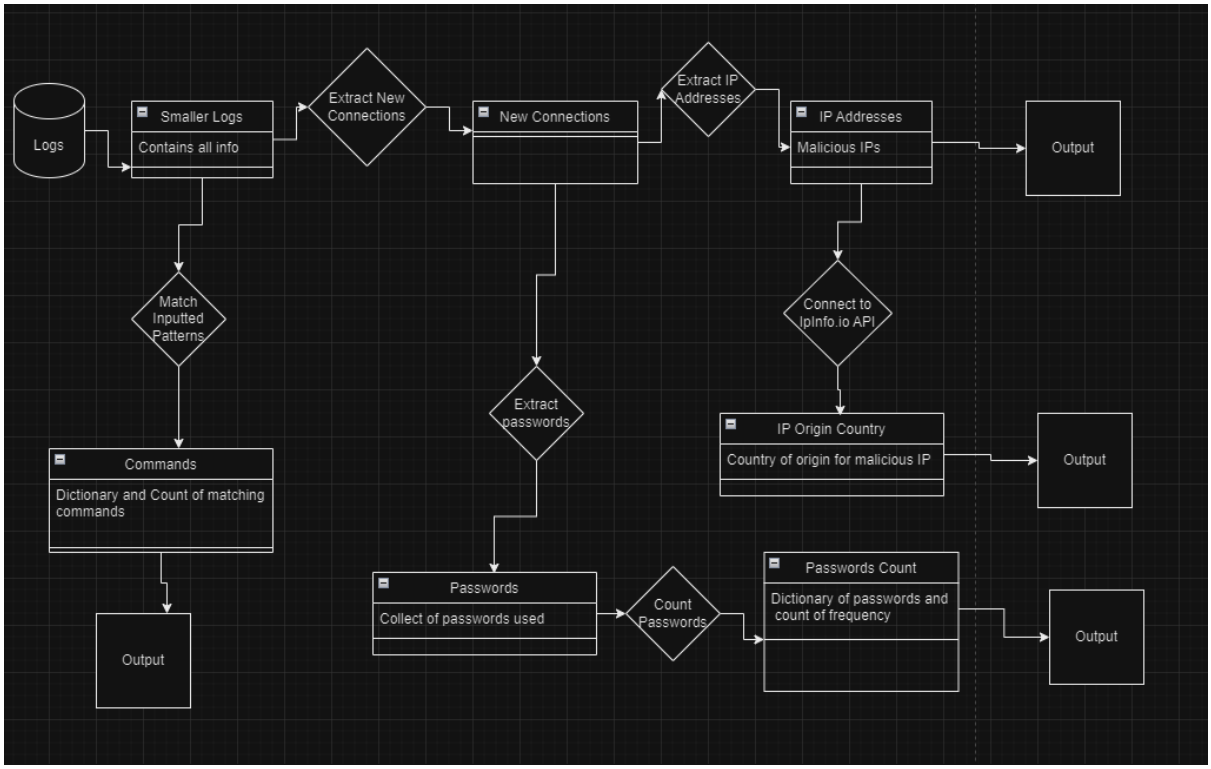
4.2 Cowrie

Cowrie will be used to implement the honeypot. Cowrie is a medium interaction SSH and Telnet honeypot which is used to log brute force attempts and shell interactions performed by an attacker. A factor in selecting Cowrie was because it is open source and allows for customisation. In additional when implemented it runs a virtual environment on the host system which will add an additional layer of protection unlike some honeypots which are run as applications. Other honeypots that were reviewed have since deprecated and or had insufficient logging capabilities. The different honeypots will have different configurations such as blocked passwords, fake directories and fake files created. To increase the quality of the data two honeypots will be configured to allow all passwords and logins. One will be configured to block the most common passwords. To prevent malicious actors from spotting

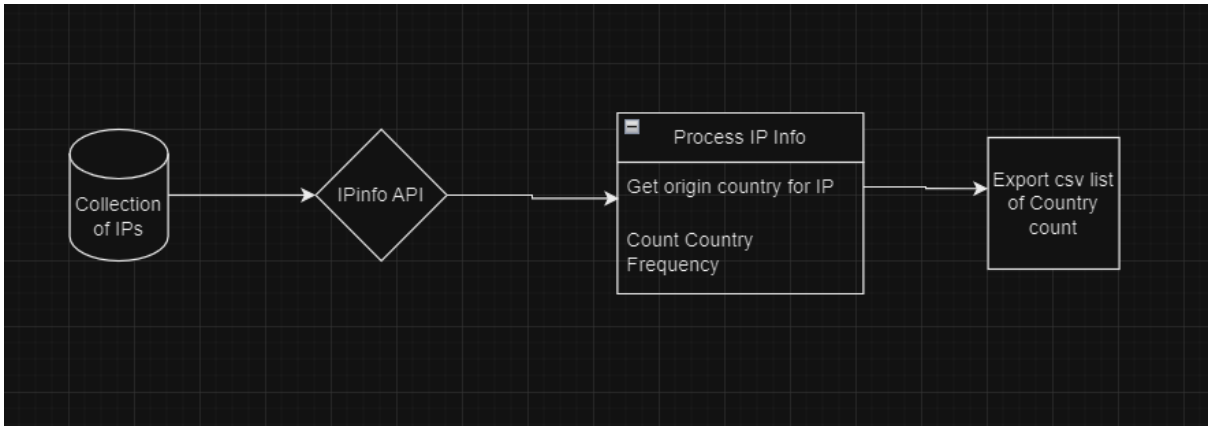
the signs that they are connected to a cowrie service, the system configurations will be changed to appear as later versions of a Ubuntu server and a personal device.

4.3 Logs

The logs are collected and processed for data analysis.



The logs are used to create a collection of malicious IP addresses. At the end a text file is created which shows each IP address that connected and a count of how frequently. With this information we can also discover the country of origin for the IP address used by the malicious actor.



From the logs the passwords are gathered to compile a list of compromised or weak passwords. This information will be built upon to provide insight on possible attack vectors. Entries in the logs that do not involve logins or passwords will be commands entered by the threat actors.

The logs are transferred to a virtual machine where they are scanned to ensure there is no malware. The log files are stored on a windows 10 device and copies are stored on a external hard drive.

There was concern in the design that the logs could be compromised by a threat actor after they connect to the system. However, Cowrie operates by creating a virtual environment. This virtual environment is how the threat actor interacts with the system. They will be inside a false directory. Any commands that are run are only simulated. Any files that are transferred to the honeypot are stored in the Cowrie directory but this cannot be accessed from inside the virtual environment directory.

---

5 Implementation

5.1 Hosting

A DigitalOcean account will be created using a student account as it provided free credit. DigitialOcean calls the virtual machines ‘Droplets’. Separate projects were created for hosting the Droplets. This would allow for quick disposal of the virtual machines should there be issues and made it easier to keep track of the different configurations.

Virtual Machines

Within each project a virtual machine was created. The specifications were the following:

Ubuntu 23.10 – 2 GB RAM 50 GB Disk

5.2 Installing Cowrie

To install Cowrie the following packages needed to be installed on the virtual machines:

Git – used for downloading and installing source code

Python3 Virtualenv – used for create Python virtual environments

Libssl-dev – Development tools for OpenSSL, TLS and SSL protocols.

Libffi-dev - Development libraries for libffi

Python3-minimal – used to ensure Python 3 essentials are installed

Authbind – used to bind root to UDP and TCP ports

Build-essential – used for building software

Libpython3-dev – Library for Python3

After using the above packages to install Cowire changes made from the default configurations. The Cowrie username was changed from ‘Phil’ to make it harder for a threat actor to spot. The ports were configured to redirect to ports 2222 and 2223.

On two honeypots fake documents and directories were created. These involved documents with names such as passwords, invoices or other sensitive information to entice a threat actor to navigate the honeypot.

On one honeypot the default passwords and top 100 most common passwords were changed and/or blocked.

Once Cowrie was launched the service was tested by seeing if I could connect to the virtual machine and what passwords were accepted or not accepted.

5.3 Honeypot in action

When the Cowrie service has been started the virtual machine will accept SSH and Telnet connections on ports 2222 and 2223. When a connection is attempted the honeypot will log the IP address, username and password. Successful connections will present the threat actor with a bash shell that replicates a Linux machine. Any commands taken within the honeypot will also be logged. Cowrie will attempt to mimic the commands of most Linux machines such as changing directory or accessing files. However, as it is a simulation, more complicated commands will not result is any action but the log files will still record the attempts. Unsuccessful attempts to connect will log the IP address, time, login name and password but refuse access to the bash shell.

5.4 Log Collection

The logs from Cowrie contain data collected over 24 hour period. They timestamp and record the interactions with the honeypot.

5.5 Gathering Logs

The analysis is performed on a Windows device so WinSCP is used to collect the logs. It allows for the Linux directories to be traversed which will be easier for locating the logs and downloading them.

5.6 Processing

Python is used for processing the log files as it is quick and simple to use and has a wide range of libraries and packages available. Anaconda and Jupyter notebook were used for creating and running Python. The following packages are used during the process:

re – it is regex module used for matching patterns

os – allows for interacting with operating system directories

requests – used for making http requests

5.7 API

ipinfo.io – this api is used to gather info about IP addresses.

5.8 Log File Breakdown

The log files can vary in size from 0.1 MB to 100 MB. In this format they are too large for processing. Using python and the os package the log files are broken into smaller parts.

The log files are quite large and will cause any Python programs to crash or reach index limits. To begin processing the logs they need to be broken down. A Python script would copy the content of a log file and write a smaller copy of the log file broken into parts of 1000 lines. The smaller files were a manageable size for the other scripts to manage.

5.9 IP Addresses

A Python script is used to collect the successful logins and create a collection of IP addresses. These IP's should be considered malicious , this is verified using abuseipdb or virustotal.

The code imports the log in a text format and creates entries in an array where each entry from the logs is a separate index. A regex will search for a pattern that matches an IP address. The matches will be added to another array that will then be used for counting the frequency. To count the frequency of the IP's a dictionary is created containing the addresses as Keys. The corresponding key value will be a count of each time the IP is found when iterating through the array of log entries. Once completed the dictionary will be outputted to a text file.

To measure the frequency of attacks originating from IP addresses the log is fed to an array using Python. The indexes that contain 'New Connections' are extracted and placed into a new Array. We then iterate through the array looking for patterns that match IP addresses. The IP addresses are placed into a dictionary. The dictionary will contain a unique set of IP addresses. Each address will be the dictionary Key and the value will be a count of how often the IP address connects. The key value initial starts as one. To gain the count the array will be iterated through again and when a match is found in the dictionary the key value will be increased by one.

At the end a text file is created of the dictionary which shows each IP address that connected and a count of how frequently.

5.10 Country Of Origin

To gather information about the origin country of the IP addresses the IP is fed into the ipInfo.io api. The python script takes the IP address and uses the requests package to connect to the ipinfo api. The script then returns a Country code which provides the origin of the malicious IP.

5.11 Passwords

The logs will also be used to harvest the passwords used to access the honeypot.

Using the packages re and os the log files are imported into an array and then using regex the array is iterated through and the passwords are placed into a dictionary. The dictionary is used to create a count of how frequently the passwords are used.

5.12 Commands

The commands entered by the threat actors in the logs will be extracted and analysed to create a better understanding of the potential attack vector.

---

6 Evaluation

6.1 Connections

The purpose of the research is to gather intelligence on malicious actors. In total 2.02 GB of logs were collected for analysis.

- 215 MB from Honeypot A
- 1.5 GB from Honeypot B
- 309 MB from Honeypot C

In examining the logs the most striking statistic is the volume of brute force attacks. While the machines were not publicly broadcasted or advertised, they were still seen as targets. The below table shows the total number of connections for each honeypot. A connection was recorded whether the login was successful or not. Each new session with a honeypot would create a 'new connection'. These new connections are what was logged and recorded.

Honeypot A - 128.199.88.89	Honeypot B - 167.71.232.56	Honeypot C - 209.38.16.42
Top 100 passwords blocked.	**Fake directory and modifications to default system.**	**Dummy files created in the directory.**
Total number of connections to each system in 30 days		
119391	**504852**	**101382**

6.2 Most Frequent Connections

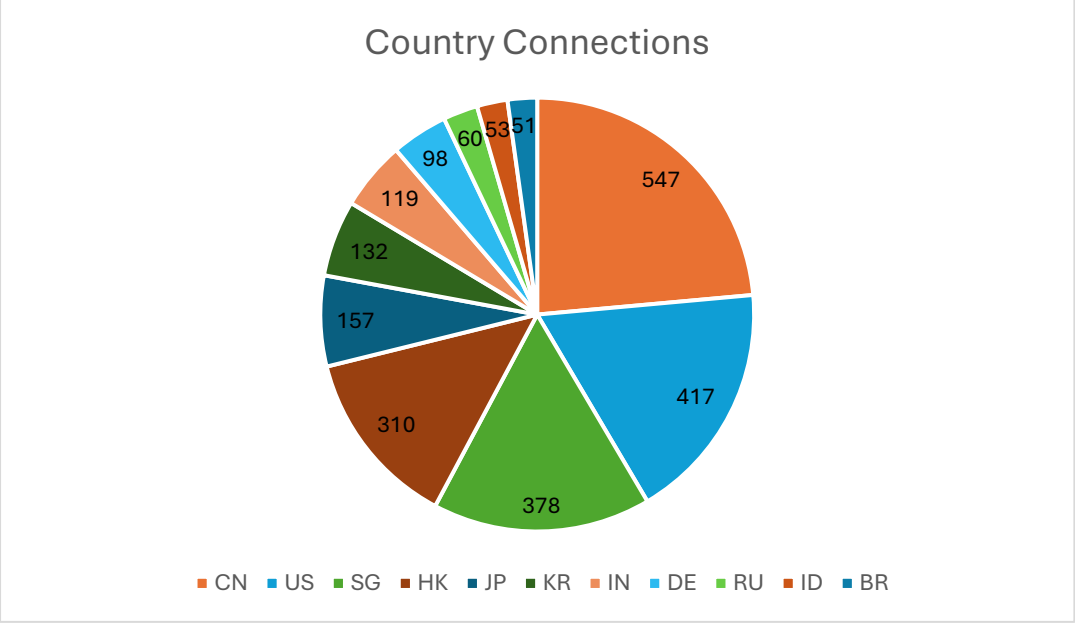
Each connection would contain information about the IP address the connection was originating from. The below table shows the top ten most frequent IP addresses and the number of connections to each honeypot.

Honeypot - 128.199.88.89		**Honeypot - 167.71.232.56**		**Honeypot - 209.38.16.42**	
IP Address	Number of Connections	**IP Address**	Number of Connections	**IP Address**	Number of Connections
137.184.226.118	23274	**167.71.228.234**	435466	**209.38.218.172**	27101
147.139.140.172	11742	**34.100.251.17**	24593	**120.78.75.128**	25755
112.90.182.230	5556	**157.173.202.91**	7930	**47.237.27.243**	13395
39.105.20.1	5206	**43.239.111.20**	6843	**124.109.53.102**	10538
120.27.249.1	4673	**123.56.126.166**	6001	**170.64.185.217**	1260
120.26.86.123	4360	**170.64.185.217**	1800	**79.137.206.11**	940
170.64.185.217	3240	**31.129.106.85**	933	**49.235.180.71**	651
139.59.102.191	3100	**49.235.180.71**	651	**193.201.9.156**	558
43.134.161.57	3100	**193.201.9.156**	338	**139.199.80.137**	343
43.134.250.114	3100	**139.199.80.137**	334	**85.209.11.227**	285

Only one address appeared in the top ten of each honeypot which was 170.64.185.217. Abuseipdb.com gave a confidence of abuse score of 100%. The address appears to originate from Australia and the ISP is DigitalOcean LLC.

6.3 Top country of origin

The IP addresses have a country of origin for the ISP associated with the addresses.



The below table shows the number of unique IP connections made by country of origin.

Honeypot A - 128.199.88.89		Honeypot B -		Honeypot C	
Country	IPs Connecting	**Country**	IPs Connecting	**Country**	IPs Connecting
CN	150	**CN**	196	**CN**	201
US	149	**US**	133	**US**	135
SG	130	**SG**	119	**SG**	129
HK	108	**HK**	111	**HK**	91
JP	52	**JP**	52	**JP**	53
KR	45	**KR**	43	**IN**	44
IN	41	**IN**	34	**KR**	44
DE	34	**RU**	31	**DE**	37
ID	32	**BR**	31	**RU**	29
BR	20	**DE**	27	**ID**	21

6.4 Passwords

The most frequently used passwords recorded by the log files. There were over 1000 different password taken from the login attempts. The below table demonstrates the top ten most used passwords. It would be advisable for a company or organization to block the use of these in any password policies.

Honeypot A - 128.199.88.89		Honeypot B - 167.71.232.56		Honeypot C -209.38.16.42	
Top 100 passwords blocked.		**Fake directory and modifications to default system.**		**Dummy files created in the directory.**	
'345gs5662d34'	424	**'345gs5662d34'**	840	**'3245gs5662d34'**	388
'3245gs5662d34'	424	**'3245gs5662d34'**	839	**'345gs5662d34'**	388
'123456'	134	**'broadguam1'**	42	**'broadguam1'**	94
'root1234567890'	93	**'sk123456.'**	39	**'admin'**	28
'agent007'	91	**'123456'**	36	**'123456'**	28
'asdf!@#123'	91	**'test@123'**	28	**'edwin'**	24
'webmonkey'	90	**'validator'**	24	**'admin123'**	22
'tribal'	88	**'12345678'**	23	**'m1'**	22
'zj!@#$%^&'	88	**'Asd123456'**	22	**'fallahi'**	22
'p@$$w0rt'	87	**'Qq123321'**	22	**'mozam'**	22
'dxyh@#13915417585'	86	**'admin123'**	21	**'fuxi'**	22
'xmhdipc'	86	**'Root123456'**	21	**'alphama'**	22
'apache@12345'	85	**'a'**	20	**'vimal'**	22
'alpin'	84	**'Aa123123'**	20	**'mahbub1'**	22
'you'	84	**'Aa123456.'**	20	**'rahim'**	22
'test@'	83	**'1234'**	19	**'ander'**	22
'HuaWei@123#'	83	**'Qq123123'**	19	**'wukong'**	22
'hbgug7800'	83	**'Aa123456!'**	18	**'cup'**	22
'Yjlgjwdata@wlzx.com'	82	**'video'**	16	**'puestovo'**	22
'ABCd^1234'	82	**'x'**	15	**'qingfuzhu'**	22
'HUAWEI123456'	82	**'grafana'**	15	**'poorya'**	22
'Asdf1234'	82	**'123123'**	15	**'yonesfar'**	22
'qwer123.com'	81	**'elango'**	15	**'santlal'**	22
'passw0RD'	81	**'pwcanswers'**	15	**'talib'**	22
'splunk123'	80	**'colucci'**	15	**'12345678'**	21

6.5 Attack Vector

The actions of the malicious connections can be broken into these categories:

- Information Gathering
- Maintaining Access

The following commands can be classified as information gathering. They are actions that collect information and specifications about the system.

6.6 *X6F\x6b - OK*

In a number of connections, the following command was run:

```
2024-06-01T07:54:29.044203Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' authenticated with b'password'
2024-06-01T07:54:29.044444Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-connection'
2024-06-01T07:54:29.295333Z [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
2024-06-01T07:54:29.295679Z [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2024-06-01T07:54:29.569112Z [twisted.conch.ssh.session#info] Executing command "b'echo -e "\\x6F\\x6B"'"
2024-06-01T07:54:29.569979Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,87539,157.173.202.91] CMD: echo -e "\x6F\x6B"
2024-06-01T07:54:29.570441Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,87539,157.173.202.91] Command found: echo -e \x6F\x6B
2024-06-01T07:54:29.570823Z [twisted.conch.ssh.session#info] exitCode: 0
2024-06-01T07:54:29.571026Z [cowrie.ssh.connection.CowrieSSHConnection#debug] sending request b'exit-status'
2024-06-01T07:54:29.571154Z [cowrie.ssh.connection.CowrieSSHConnection#info] sending close 0
2024-06-01T07:54:29.823436Z [cowrie.ssh.session.HoneyPotSSHSession#info] remote close
2024-06-01T07:54:29.823885Z [HoneyPotSSHTransport,87539,157.173.202.91] Closing TTY Log:
var/lib/cowrie/tty/eafdc691c2945a067fa5de7bac393326241395a9cd11bc6737c7191859f13b80 after 0 seconds
```

CMD: echo -e "\x6F\x6B" Command found: echo -e \x6F\x6B

The command is a bash command that sends the ashii characters ‘OK’ to the terminal. It would appear the goal is to test that commands can be run on the system. After the command is successfully run the connection ends.

6.7 *Uname*

The command ‘uname’ was run in an attempt to gather information about the system.

```
2024-06-11T15:46:31.286415Z [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2024-06-11T15:46:31.478631Z [twisted.conch.ssh.session#info] Executing command "b'uname -s -v -n -r -m'"
2024-06-11T15:46:31.479476Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,303661,170.64.185.217] CMD: uname -s -v -n -r -m
2024-06-11T15:46:31.479867Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,303661,170.64.185.217] Command found: uname -s -v -n -r -m
2024-06-11T15:46:31.480228Z [twisted.conch.ssh.session#info] exitCode: 0
2024-06-11T15:46:31.480332Z [cowrie.ssh.connection.CowrieSSHConnection#debug] sending request b'exit-status'
```

CMD: uname -s -v -n -r -m

Commands containing uname were run the following amount of times:

Honeypot A - 128.199.88.89	Honeypot B - 167.71.232.56	Honeypot C -209.38.16.42

Top 100 passwords blocked.	Fake directory and modifications to default system.	Dummy files created in the directory.
326	4248	4260

Variations of the command were found which gathered different data:

unamne -a : Retrieves and prints all information on the system.

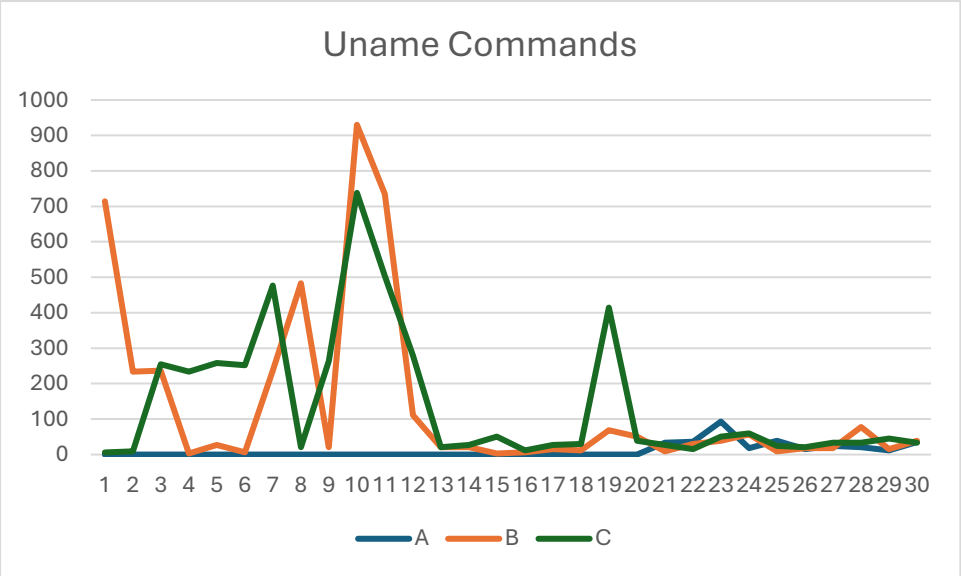
uname -s : Prints the kernel name.

uname -v : Prints the kernel version.

uname -n : Prints the network node hostname.

uname -r : Prints the kernel release.

uname -m : Prints the machine hardware name.



Over the 30 day period there appears to be a decline in instances of gathering system information. Honeypots B and C appear to have similar spikes near the 7th and 8th day and later near the 10th and 11th days.

6.8 Busybox

These commands appear to be an attempt to extract information from the system using a Unix suite called Busybox.

```
CMD: enable
Command found: enable
Reading txtcmd from "share/cowrie/txtcmds/bin/enable"
CMD: system
Can't find command system
Command not found: system
```

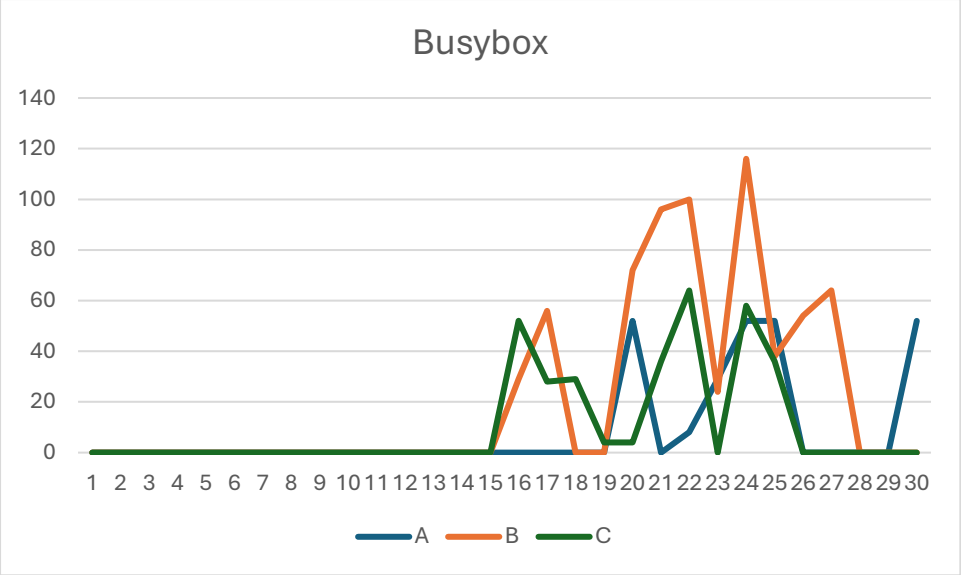
```
CMD: shell
Can't find command shell
Command not found: shell
CMD: sh
Command found: sh
CMD: ping; sh
Command found: ping
Command found: sh
CMD: busybox dd if=$SHELL bs=22 count=1||dd if=/proc/self/exe bs=22 count=1||while read i;do
busybox echo -n $i;done</proc/self/exe||cat /proc/self/exe
Command found: busybox dd if=$SHELL bs=22 count=1
Command found: dd if=$SHELL bs=22 count=1
Command found: dd if=/proc/self/exe bs=22 count=1
Can't find command while
Command not found: while read i
Command found: do busybox echo -n
Command found: done < /proc/self/exe
Command found: cat /proc/self/exe
```

The **enable** command is frequently used in devices such as Cisco to elevate privileges. The attacker may then be using System to gain configurations or system level functions. The **shell** command is not a typical Linux command. The attacker is probably trying to gain a shell interface. The **sh** command is to attempt to access a unix shell. The **ping** command is an attempt to check network interfaces.

The next command is a collection of commands that uses **Busybox** which combines command Unix utilities into a single executable. The **dd** command is used for copying and converting files. The next part of the command is for attempting to copy data from a shell binary **if=$SHELL bs=22 count=1**.

The final part of the command **/proc/self/exe** attempts to execute the binary of the process and read the contents.

Honeypot A - 128.199.88.89	Honeypot B - 167.71.232.56	Honeypot C -209.38.16.42
Top 100 passwords blocked.	**Fake directory and modifications to default system.**	**Dummy files created in the directory.**
245	649	311



From the above graph we see that there is an increase in these commands towards the end of the 30 month period.

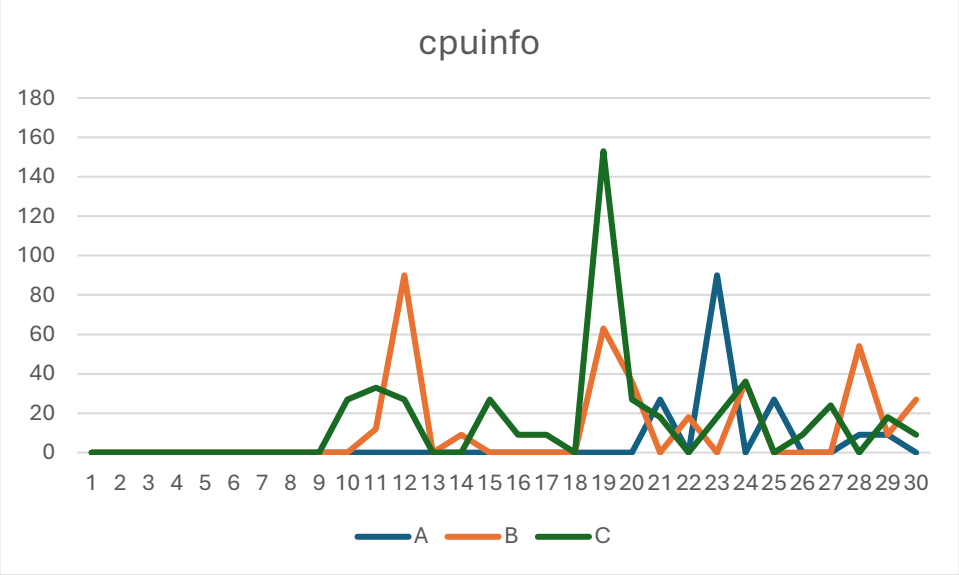
6.9 CPUINFO

```
2024-06-19T02:36:41.421935Z [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2024-06-19T02:36:41.743032Z [twisted.conch.ssh.session#info] Executing command "b"cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'""
2024-06-19T02:36:41.744365Z [SSHChannel session (5) on SSHService b'ssh-connection' on HoneyPotSSHTransport,133843,115.231.13.16] CMD: cat /proc/cpuinfo | grep name |
head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'
2024-06-19T02:36:41.745090Z [SSHChannel session (5) on SSHService b'ssh-connection' on HoneyPotSSHTransport,133843,115.231.13.16] Command found: awk {print $4,$5,$6,
$7,$8,$9;}
2024-06-19T02:36:41.745217Z [SSHChannel session (5) on SSHService b'ssh-connection' on HoneyPotSSHTransport,133843,115.231.13.16] Command found: head -n 1
2024-06-19T02:36:41.745302Z [SSHChannel session (5) on SSHService b'ssh-connection' on HoneyPotSSHTransport,133843,115.231.13.16] Command found: grep name
2024-06-19T02:36:41.745380Z [SSHChannel session (5) on SSHService b'ssh-connection' on HoneyPotSSHTransport,133843,115.231.13.16] Command found: cat /proc/cpuinfo
2024-06-19T02:36:41.746376Z [twisted.conch.ssh.session#info] exitCode: 0
2024-06-19T02:36:41.746552Z [cowrie.ssh.connection.CowrieSSHConnection#debug] sending request b'exit-status'
```

CMD: cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'

The command **cat /proc/cpuinfo** is used to display the CPU information. The **grep name** command will filter for lines containing the word name. If there is a matching line the command **head -n 1** will select it. The following command awk '{print $4,$5,$6,$7,$8,$9;}' extracts specific fields.

Honeypot A - 128.199.88.89	Honeypot B - 167.71.232.56	Honeypot C -209.38.16.42
Top 100 passwords blocked.	**Fake directory and modifications to default system.**	**Dummy files created in the directory.**
162	354	444



6.10 Maintaining Access

The below commands are classified as attempts by the threat actors to maintain access to the system.

6.11 Secure.sh and PKILL

The following commands appear to be an attempt to maintain access to the system.

```
2024-06-19T17:33:58.345303Z [HoneyPotSSHTransport,134419,20.219.21.194] login attempt [b'345gs5662d34'/b'345gs5662d34'] failed
2024-06-19T17:33:58.583036Z [twisted.conch.ssh.session#info] Executing command "b'rm -rf /tmp/secure.sh; rm -rf /tmp/auth.sh; pkill -9 secure.sh; pkill -9 auth.sh;
echo > /etc/hosts.deny; pkill -9 sleep;'"
2024-06-19T17:33:58.584243Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] CMD: rm -rf /tmp/secure.sh; rm -rf
/tmp/auth.sh; pkill -9 secure.sh; pkill -9 auth.sh; echo > /etc/hosts.deny; pkill -9 sleep;
2024-06-19T17:33:58.584910Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: rm -rf /tmp/secure.sh
2024-06-19T17:33:58.585153Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: rm -rf /tmp/auth.sh
2024-06-19T17:33:58.585547Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: pkill -9 secure.sh
2024-06-19T17:33:58.585638Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Reading txtcmd from
"share/cowrie/txtcmds/usr/bin/pkill"
2024-06-19T17:33:58.586086Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: pkill -9 auth.sh
2024-06-19T17:33:58.586192Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Reading txtcmd from
"share/cowrie/txtcmds/usr/bin/pkill"
2024-06-19T17:33:58.586361Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: echo > /etc/hosts.deny
2024-06-19T17:33:58.587026Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Command found: pkill -9 sleep
2024-06-19T17:33:58.587136Z [SSHChannel session (4) on SSHService b'ssh-connection' on HoneyPotSSHTransport,134409,36.26.71.117] Reading txtcmd from
"share/cowrie/txtcmds/usr/bin/pkill"
2024-06-19T17:33:58.587356Z [twisted.conch.ssh.session#info] exitCode: 0
```

Executing command "b'rm -rf /tmp/secure.sh; rm -rf /tmp/auth.sh; pkill -9 secure.sh; pkill -9 auth.sh; echo > /etc/hosts.deny; pkill -9 sleep;'"
CMD: rm -rf /tmp/secure.sh; rm -rf /tmp/auth.sh; pkill -9 secure.sh; pkill -9 auth.sh; echo > /etc/hosts.deny; pkill -9 sleep;
Command found: rm -rf /tmp/secure.sh
Command found: rm -rf /tmp/auth.sh
Command found: pkill -9 secure.sh
Reading txtcmd from "share/cowrie/txtcmds/usr/bin/pkill"
Command found: pkill -9 auth.sh
Reading txtcmd from "share/cowrie/txtcmds/usr/bin/pkill"
Command found: echo > /etc/hosts.deny

Command found: pkill -9 sleep Reading txtcmd from "share/cowrie/txtcmds/usr/bin/pkill"

rm -rf /tmp/secure.sh: This command removes the file secure.sh from the temporary /tmp directory.

rm -rf /tmp/auth.sh: Removes the file auth.sh from the /tmp directory.

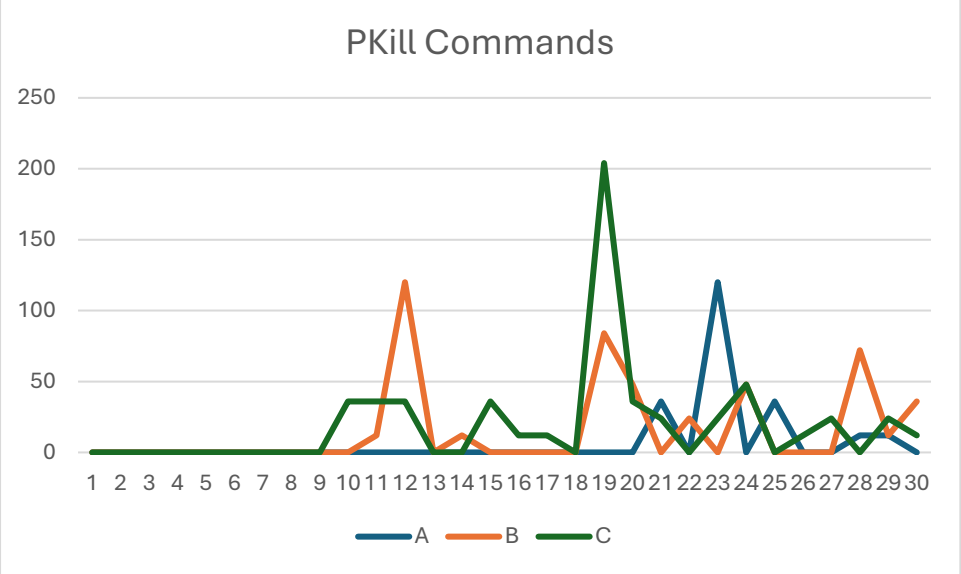
pkill -9 secure.sh: Forcefully terminates any processes that are named secure.sh.

pkill -9 auth.sh: This forcefully terminates any processes named auth.sh.

echo > /etc/hosts.deny: Clears the contents of the /etc/hosts.deny file, which could be used to block specific hosts from accessing the system.

pkill -9 sleep: Forcefully terminates any sleep processes which prevents the system from shutting down.

These commands appear to be attempts to remove or cover up a scripts or services. The hosts.deny could be an attempt to allow broader access to the system while pkill will prevent the system from disconnecting.



Honeypot A - 128.199.88.89	Honeypot B - 167.71.232.56	Honeypot C -209.38.16.42
Top 100 passwords blocked.	**Fake directory and modifications to default system.**	**Dummy files created in the directory.**
216	468	576

6.12 chmod +x setup.sh

To maintain access to the device the threat actor attempts to establish a foothold by altering the authorised keys file. The below command was detected 282 times.


```
Command found: chmod +x setup.sh
Command found: sh setup.sh
Command found: rm -rf setup.sh
Command found: mkdir -p ~/.ssh
Command found: chattr -ia
Command found: echo ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCqHrvnL6l7rT/mt1AdgdY9tC1GPK216q0q/7neN
Vqm7AgvfJIM3ZKniGC3S5x6KOEApk+83GM4IKjCPfq007SvT07qh9AscVxegv66I5yuZTEaD
AG6cPXxg3/0oXHTOTvxelgbRrMzfU5SEDAEi8+ByKMefE+pDVALgSTBYhol96hu1GthAMtP
AFahqxrvaRR4nL4ijxOsmSLREoAb1lxiX7yvoYLT45/1c5dJdrJrQ60uKyieQ6FieWpO2xF6tzfdm
HbiVdSmdw0BiCRwe+fuknZYQxIC1owAj2p5bc+nzVTi3mtBEk9rGpgBnJ1hcEUslEf/zevIcX8+6
H7kUMRr rsa-key-20230629 > ~/.ssh/authorized_keys
Command found: chattr +ai ~/.ssh/authorized_keys
```

With **chmod +x setup.sh** the threat actor is changing the permissions of setup.sh to make it executable.

The **sh setup.sh** command runs the setup script and to cover the execution they run the following command to remove the script once it is executed.

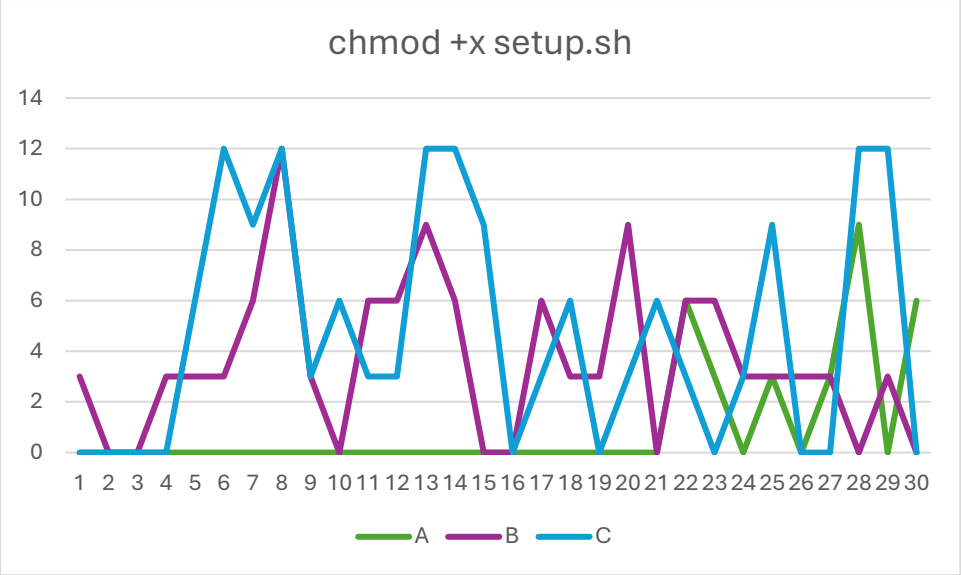
This will create a ssh directory: **mkdir -p ~/.ssh**

Then the threat actor runs: **chattr -ia ~/.ssh/authorized_keys**

This is to remove the immutable and append-only attributes from the authorized_keys file, allowing modifications.

After that the threat actor adds a new SSH public key to the authorized_keys file. This allows the attacker to access the machine via SSH using this key.

Finally they run this command to prevent further modifications: **chattr +ai ~/.ssh/authorized_keys:**



6.13 Files Collected

Attempts to further establish control was seen by the transfer of a file to the honeypot devices.

```
2024-06-01T12:14:35.656861Z [cowrie.ssh.session.HoneyPotSSHSession#info] remote close
2024-06-01T12:14:35.657322Z [HoneyPotSSHTransport,94448,78.153.140.51] Saved redir contents with SHA-256
8a68d1c08ea31250063f70b1ccb5051db1f7ab6e17d46e9dd3cc292b9849878b to var/lib/cowrie/downloads/8a68d1c08ea31250063f70b1ccb5051db1f7ab6e17d46e9dd3cc292b9849878b
2024-06-01T12:14:35.657886Z [HoneyPotSSHTransport,94448,78.153.140.51] Closing TTY Log:
var/lib/cowrie/tty/96abae0475aed33d163866113bf441296b0f7de7c3175e634e29a5b0f5aa4014 after 0 seconds
2024-06-01T12:14:35.663144Z [HoneyPotSSHTransport,94448,78.153.140.51] SFTP Uploaded file "redtail.arm7" to
var/lib/cowrie/downloads/2be800f792d9dfea4e5644b3c340f193568126b4771e0c2dcb95e0d047464b41
2024-06-01T12:14:35.668684Z [HoneyPotSSHTransport,94448,78.153.140.51] SFTP Uploaded file "redtail.arm8" to
var/lib/cowrie/downloads/b9566789c853f706dc06e947eb3d19ce7859c3483f6e7e85296b28f4a8e9090d
2024-06-01T12:14:35.676358Z [HoneyPotSSHTransport,94448,78.153.140.51] SFTP Uploaded file "redtail.i686" to
var/lib/cowrie/downloads/eb3b0390f06a0c13383c7478f4f1a55520a31b8668141b3b2792c371e7bcba69
2024-06-01T12:14:35.685081Z [HoneyPotSSHTransport,94448,78.153.140.51] SFTP Uploaded file "redtail.x86_64" to
var/lib/cowrie/downloads/8c8d832581a492083e8a97a1016a4ce86a3e0f0c20b21d21e6334e47982719bb
2024-06-01T12:14:35.685758Z [HoneyPotSSHTransport,94448,78.153.140.51] SFTP Uploaded file "setup.sh" to
var/lib/cowrie/downloads/5ffdf7536899526ec78197a286399f011f4723f814412d097aa65d76072f1b65
2024-06-01T12:14:35.686162Z [HoneyPotSSHTransport,94448,78.153.140.51] avatar root logging out
2024-06-01T12:14:35.686241Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-06-01T12:14:35.686301Z [HoneyPotSSHTransport,94448,78.153.140.51] Connection lost after 42 seconds
2024-06-01T12:14:35.688460Z [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
```

- Saved redir contents with SHA-256 8a68d1c08ea31250063f70b1ccb5051db1f7ab6e17d46e9dd3cc292b9849878b to var/lib/cowrie/downloads/8a68d1c08ea31250063f70b1ccb5051db1f7ab6e17d46e9dd3cc292b9849878b
- Closing TTY Log: var/lib/cowrie/tty/96abae0475aed33d163866113bf441296b0f7de7c3175e634e29a5b0f5aa4014 after 0 seconds
- SFTP Uploaded file "redtail.arm7" to var/lib/cowrie/downloads/2be800f792d9dfea4e5644b3c340f193568126b4771e0c2dcb95e0d047464b41
- SFTP Uploaded file "redtail.arm8" to var/lib/cowrie/downloads/b9566789c853f706dc06e947eb3d19ce7859c3483f6e7e85296b28f4a8e9090d

- Uploaded file "redtail.i686" to var/lib/cowrie/downloads/eb3b0390f06a0c13383c7478f4f1a55520a31b8668141b3b2792c371e7bcba69
- SFTP Uploaded file "redtail.x86_64" to var/lib/cowrie/downloads/8c8d832581a492083e8a97a1016a4ce86a3e0f0c20b21d21e6334e47982719bb
- SFTP Uploaded file "setup.sh" to var/lib/cowrie/downloads/5ffdf7536899526ec78197a286399f011f4723f814412d097aa65d76072f1b65

Multiple files were uploaded to the honeypot using SFTP.

Malicious Files Collected

The files uploaded to the honeypots were gathered and placed on a sandbox environment to ensure that the researchers device was protected from infection. The files were uploaded to VirusTotal to test if they were malicious. Below is the Sha 256 number for files that were flagged as malicious.

Sha256	**Verdict**	**Rating**
01ba4719c80b6fe911b091a7c05124b64eeece964e09c058ef8f9805daca546b	Not Malicious	0/64
2b430c74a64f241e74b5329ff2d1fe3127abde7a10b6acb66cf5188fa29e1d20	Malicious	2/67
9ef2ef02376445bf4c145820c0c81f2bbe0b96f2017278562e0bd259bf7bd061	Malicious	33/64
ab897157fdef11b267e986ef286fd44a699e3699a458d90994e020619653d2cd	Malicious	33/67
ea9f3911ff2884621874c1e98b5dc9139964adeab333b92816eb5c307d73a67f	Malicious	31/65
d46555af1173d22f07c37ef9c1e0e74fd68db022f2b6fb3ab5388d2c5bc6a98e	Not Malicious	0/63

6.14 Discussion

The research has provided a better understanding of how a cloud hosted system is infiltrated by threat actors. With 725625 connections made to the honeypots over the 30 period it appears that bots are continuously scanning IP addresses and attempting to make connections. Showing that any exposed system cannot be left without a firewall or some form of defence in place. The report has shown that blocking common passwords can potentially reduce the number of unauthorised accesses significantly. In the honeypot where passwords were blocked there was 119391 connections, compared to 504852 on the second system. Although this would require further testing as the third honeypot recorded the least connections with 101382.

A pattern appeared to emerge in the logs where after successfully making the connection, the threat actors would begin to gather information about the systems. The honeypots where the default system names were changed recorded a higher number of requests for system information. It would suggest that when the default name provided by the Cowrie honeypot was detected the malicious actors would opt not to waste any further time and resources on a simulated system. The immediate disconnection seems to suggest the connections were made by bots. Further evidence of this was how the fake files and directories did not appear to have been interacted with. The commands like ‘uname’ seemed to occur more frequently at the beginning of the month while the more complicated commands appeared towards the end. It is unclear at the moment whether that is because botnets do this deliberately as part of a pattern to gathering information or if there are more resources available to malicious actors near the end of a month. Further research would be needed. Another unusual pattern that formed was how the top two passwords '345gs5662d34' and '3245gs5662d34' were used on each of the honeypots although the order was flipped on the third honeypot. The number of logins for the first and second login was almost identical then there was a significant drop in number of connections. This could be explained that the connections with '345gs5662d34' and '3245gs5662d34' monopolised the time and prevented other connections occurring as frequently.

The research provided insights into how the threat actors would attempt to maintain access. Once entry was gained to the system, they would alter the ssh-rsa file to ensure that their ssh key fingerprint would have continuous access. The file would be copied to a new directory called ssh and a file created called authorised_keys. With this information efforts should be made to lock down access to where ssh keys are stored on systems.

Malicious files were also discovered on the honeypots. Collecting the Sha256 number allows for organizations to search for or detect their appearance on devices or networks. The files seemed to be for remote access but further research would be needed in an isolated environment to confirm. Other information gathered that could be used to strengthen an organisations security posture included the collection of compromised passwords and a large number of malicious IP addresses. These IP addresses should be blocked by an organisation’s firewalls or network defences.

Hosting with Digital Ocean proved to be a suitable way to isolate the honeypots from any sensitive systems. The was no indications of the malicious actors being able to escape the

Cowrie honeypot to access other systems. Cowrie and Digital Ocean were an efficient way for hosting the honeypots. The systems could be quickly created and were not resource heavy. They could be run using the smallest Ubuntu Linux virtual machine available. As a honeypot Cowrie was able to record the commands and most interactions of the threat actors that connected.

To manage with the 2.02 GB logs of data using Python to reduce the sizes and extract information proved effective. When the logs were left in their original size Python would crash while the scripts were being run. The scripts were useful for extracting the IP addresses, passwords and commands being used and exporting the information to text and csv files.

A limitation of the research was the number commands simulated by Cowrie. Based upon the frequency of 'uname' or '-e "\x6F\x6B"' commands were entered, it would have been advantageous to the research to have a response generated by the system. This would have allowed to further follow and examine the potential attack path by threat actors. It would also reduce the chances of the honeypot being detected by returning expected commands. A possible limitation that was unnoticed until the logs were collected was that the IP addresses only appeared to be IPBV4. It would require further research to see if Cowrie does not record IPBV6 addresses or whether are they just less common. An area the research should have been expanded to include was the username. While blocking the most common passwords provided an insight into how the threat actors would respond to a basic password policy being in place a similar approach should have been implemented with the user names. A majority of the logins were using the 'root' account. Blocking 'root' and other common usernames such as 'Admin' or 'Administrator' could have yielded interesting results, Another way the research could have been improved would have been to open additional ports up to see if there is an increase in attacks or different behaviour recorded. The honey traps were only configured to accept connections to ports 2222 and 2223.

7 Conclusion

The honeypots were an effective way of performing cybersecurity research. They can provide information on malicious IP's and compromised passwords while also giving insight into how a threat actor will attempt to collect information and take over a system. The honeypots were only run for a 30-day period. Any future research should run for a longer period to see any new attack vectors are discovered or whether there is any different behaviour during other months of the year. A longer period might lead to the fake files and folders being interacted with. It would also be interesting to see whether the same commands occur as frequency each month following the same rise and fall pattern. Similar research honeypots should be run on a regular basis to keep up to date with the latest threats.

Bibliography

A. Subhan, Y. N. (2023). Analyzing the Attack Pattern of Brute Force Attack on SSH Port. *A. Subhan, Y. N. Kunang an2023 International Conference on Information Technology and Computing (ICITCOM)*.

al, R. A. (2022). Long-Term Study of Honeypots in a Public Cloud. *R. Agrawal et al., "Long-Term Study of Honeypots in a Pu52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*.

Ali, T. O. (2022). Introduction to Cyber Tensions Preventative Analysis and Honeypotting Strategy. *International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA)*.

Arora, N. B. (2018). Intrusion Detection Using Honeypots. *Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*.

Chandane, V. S. (2021). Efficacy Measuring Framework for the Assessment of Dynamic Honeypot. *V. Shirsath and M. M. Chandane, "Efficacy Measuring Framework for the Assessment International Conference on Advances in Computing, Communication, and Control (ICAC3)*.

Dadarlat, M. L. (2018). Honeypot in the cloud Five years of data analysis. *17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*.

Georgieva, A. B. (2020). Log Files Analysis For Network Intrusion Detection. *IEEE 10th International Conference on Intelligent Systems (IS)*, (pp. pp. 328-333). Varna, Bulgaria.

Lapin, V. V. (2018). Architecture of the Honeypot System for Studying Targeted Attacks. *XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*.

M. Başer, E. Y. (2021). SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot. *M. Başer, E. Y. Güven and M. A. Aydı6th International Conference on Computer Science and Engineering (UBMK)*.

Mittal, M. G. (2024). Review for Prevention of Botnet Attack Using Various Detection Techniques in IoT and IIoT. *2nd International Conference on Disruptive Technologies (ICDT)*, (pp. pp. 259-264). Greater Noida, India.

P. S. Negi, A. G. (2020). P. S. Negi, A. Garg and R. LalIntrusion Detection and Prevention using Honeypot Network for Cloud Security. *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*.

S. Dowling, M. S. (2017). Using analysis of temporal variances within a honeypot dataset to better predict attack type probability. *S. Dowling, M. Schukat and H. Melv12th International Conference for Internet Technology and Secured Transactions (ICITST)*.

S. Sunil, A. S. (2023). Log Based Anomaly Detection: Relation Between The Logs. *International Conference on Networking and Communications (ICNWC)* (pp. pp. 1-5). Chennai, India: doi: 10.1109/ICNWC57852.2023.10127571.