

# EN4720: Security in Cyber-Physical Systems

## Exercise — Authentication

Name: Thalagala B. P.

Index No: 180631J

May 14, 2023

**This is an individual exercise!**  
**Due Date: 19 May 2023 by 11.59 PM**

Content adopted from the Udacity Security Engineer Nano-Degree and ECU CSI 1101.

This exercise has to be carried out using a Linux-based PC/virtual machine. Read all the instructions and questions before attempting the exercise. Add answers under each question in the Questions section and submit the resulting PDF. All files used in this exercise can be found in *ex3-resources* folder.

### Instructions

1. Explore and understand how Unix systems store and manage passwords.
2. Download the shadow.hex file from the “files” folder. This file has been encrypted using RC4 encryption. 40 bits hexadecimal key: **5D 49 34 71 64**. Decrypt the file using any available decryption tool (e.g., cryptool).
3. Download the “500\_passwords.txt” file from the “files” folder. Run “hashcat” password cracking utility to crack the passwords in the decrypted shadow file with the help of the dictionary “500\_passwords.txt”. Please note that, to crack the passwords in the decrypted shadow file using hashcat, you need to take encrypted password fields of each user into a different file. You may rename that file as “encrypted\_passwords.txt”.
4. Download the “myhashes.txt”, “mydictionary.txt”, “myruleset.txt” files from the “files” folder. Note that, myhashes.txt contains MD5 hashes. Run a rule-based attack using “hashcat” password cracking utility to crack the passwords in the myhashes.txt file with the help of the dictionary “mydictionary.txt” and the ruleset myruleset.txt.
5. Answer the questions given below.

## Questions

1. What is the difference between `/etc/passwd` and `/etc/shadow` files? Complete Table 1.

*`/etc/passwd` file stores the account details of the users on a given host machine. The content of the file is word-readable and everyone has the read permission for that file.*

```
(user@host)-[/etc]
$ ls -lh | grep passwd
-rw-r--r-- 1 root root 2.0K Apr 2 14:44 passwd
-rw-r--r-- 1 root root 2.0K Apr 2 14:38 passwd-
```

*`/etc/shadow` file stores the details about the passwords of users and hashed passwords. The shadow file can only be read by the root account.*

```
(user@host)-[/etc]
$ ls -lh | grep shadow
-rw-r----- 1 root shadow 770 Apr 2 14:44 gshadow
-rw-r----- 1 root shadow 761 Apr 2 14:44 gshadow-
-rw-r----- 1 root shadow 968 Apr 2 14:44 shadow
-rw-r----- 1 root shadow 949 Apr 2 14:38 shadow-
```

Table 1: Difference between `/etc/passwd` and `etc/shadow` files

Attribute	<code>/etc/passwd</code>	<code>etc/shadow</code>
Number of fields	7	9
Accessible users	Anyone can read	Only <code>root</code> user and members of <code>shadow</code> group can read

2. You will find seven fields in the `/etc/passwd` file as indicated in Figure 1. Explain the purpose of each field.

`games:x:5:60:games:/usr/games:/usr/sbin/nologin`

The diagram illustrates the fields of the `games` entry in the `/etc/passwd` file. The entry is `games:x:5:60:games:/usr/games:/usr/sbin/nologin`. The fields are separated by colons. The labels and their corresponding values are: User ID (5), UID (5), GID (60), Encrypted Password (x), User Description (games), User's Home Directory (/usr/games), and User's Login Shell (/usr/sbin/nologin).

Figure 1: Example storage format of `/etc/passwd` file

- (a) `games`: name of the user for whom this entry corresponds to.
- (b) `x` : indicates that a password exists for the user. However, the password is stored in the `/etc/shadow` file. If instead of `x` it shows a `!` symbol, this indicates that a password does not exist.
- (c) `5` : User ID of this user. User IDs for created users begin from 1000 and run up to 59999
- (d) `60`: Group ID of the group this user belongs to.
- (e) `games`: can contain multiple comma separated sub-fields of information inclusive of full name and telephone numbers. Here, no telephone numbers have been provided.

- (f) /usr/games : location of home directory assigned to this user.
- (g) /usr/sbin/nologin : default shell assigned to this user.

**Note:** Information adopted from,

<https://www.maketecheasier.com/how-linux-stores-manages-user-passwords/>

3. Dump the content of your /etc/shadow file to the terminal using `sudo cat /etc/shadow` and add a screenshot.

```

(kalibee@LAP-BIMALKA98)-[/etc]
$ sudo cat shadow
[sudo] password for kalibee:
root:*:19428:0:99999:7:::
daemon:*:19428:0:99999:7:::
bin:*:19428:0:99999:7:::
sys:*:19428:0:99999:7:::
sync:*:19428:0:99999:7:::
games:*:19428:0:99999:7:::
man:*:19428:0:99999:7:::
lp:*:19428:0:99999:7:::
mail:*:19428:0:99999:7:::
news:*:19428:0:99999:7:::
uucp:*:19428:0:99999:7:::
proxy:*:19428:0:99999:7:::
www-data:*:19428:0:99999:7:::
backup:*:19428:0:99999:7:::
list:*:19428:0:99999:7:::
irc:*:19428:0:99999:7:::
_apt:*:19428:0:99999:7:::
nobody:*:19428:0:99999:7:::
systemd-network:!:19428:0:99999:7:::
kalibee:$y$j9T$bqi0jQs2qKqJqk4...7kH1:19449:0:99999:7:::
tss:!:19449:0:99999:7:::
strongswan:!:19449:0:99999:7:::
messagebus:!:19449:0:99999:7:::
usbmux:!:19449:0:99999:7:::
dnsmasq:!:19449:0:99999:7:::
avahi:!:19449:0:99999:7:::
speech-dispatcher:!:19449:0:99999:7:::
pulse:!:19449:0:99999:7:::
saned:!:19449:0:99999:7:::
lightdm:!:19449:0:99999:7:::
polkitd:!:19449:0:99999:7:::
rtkit:!:19449:0:99999:7:::
colord:!:19449:0:99999:7:::
nm-openvpn:!:19449:0:99999:7:::
nm-openconnect:!:19449:0:99999:7:::
xrdp:!:19449:0:99999:7:::

```

Figure 2: Content of /etc/shadow file

4. Identify the different fields present in one line in the shadow file and explain the purpose of each field. Clearly mark the fields and provide the explanation.

*there are 9 fields in an entry in the shadow file, and each field can be identified as follows. Let us consider the entry of the user “kalibee”*

- (a) kalibee - name of the user for whom this entry corresponds to
- (b) \$y\$j9T\$bqi0jQs2qKqJqk4...7kH1 - hashed user password. Along with the information about, the hashing algorithm, used salt value.
- (c) 19449 - last password change time (in days since Jan 1, 1970)
- (d) 0 - the number of days after which the password can be changed. A value of 0 means the password can be changed at any time.
- (e) 99999 - the number of days after which the password must be changed. A value of 99999 indicates a user can retain the password as long as desired.
- (f) 7 - if the password is set to expire, this fields indicates the number of days to warn the user about password expiry.

(g) ::: - Three empty fields. The first one indicates the number of days to wait after password expiry, following which the account will be disabled. The second one indicates the number of days since January 1, 1970, that an account has been disabled. The third field is reserved for future use. The empty fields indicate that the existing password for this user has not expired and is not set to expire soon.

**Note:** Information adopted from,

<https://www.maketecheasier.com/how-linux-stores-manages-user-passwords/>

5. Complete Step #2 mentioned under **Instructions**. Add a screenshot of the decrypted shadow.hex file content. What is the type of hash in the decrypted shadow file?

*Your answer here*

6. Complete Step #3 mentioned under **Instructions**. Add a screenshot of a part of the “500\_passwords.txt” file content using vi/vim/nano.

*Your answer here*

7. What is the command you used to crack the passwords in the shadow file? Briefly explain each field in the command.

*Your answer here*

8. Add a screenshot of all the cracked passwords.

*Your answer here*

9. Provide recommendations to enhance the strength of the passwords.

*Your answer here*

10. Briefly explain about “Rule-based Attack” and how it uses time and memory for the hash cracking process using a diagram.

*Your answer here*

11. Complete Step #4 mentioned under **Instructions**. Add a screenshot of the mydictionary.txt and myruleset.txt file contents. Define each rule in myruleset.txt using standard rules in the rule-based attack.

*Your answer here*

12. Add a screenshot of all the cracked passwords. Run a dictionary attack without myruleset.txt (using only the dictionary) and compare the two cracked files.

*Your answer here*