To ensure an operating system is secure, all potential security vulnerabilities that could compromise the integrity of both the system and the data it holds must be effectively isolated. Data integrity is a fundamental requirement, as is the protection of user privacy, especially for those directly interacting with the system or its applications. Achieving true security goes beyond just implementing a firewall, antivirus software, or the security measures commonly used today. A secure operating system must safeguard everything from the magnetic storage media to the data stored within it. Credentials such as passwords are sensitive and must remain confidential, inaccessible to unauthorized individuals. To protect this vital information, encryption methods are employed, ensuring that passwords and other sensitive data cannot be exploited by attackers. The process of securing operating systems, computers, mobile devices, and all connected technologies is ongoing, requiring continuous effort and vigilance to stay ahead of emerging threats.

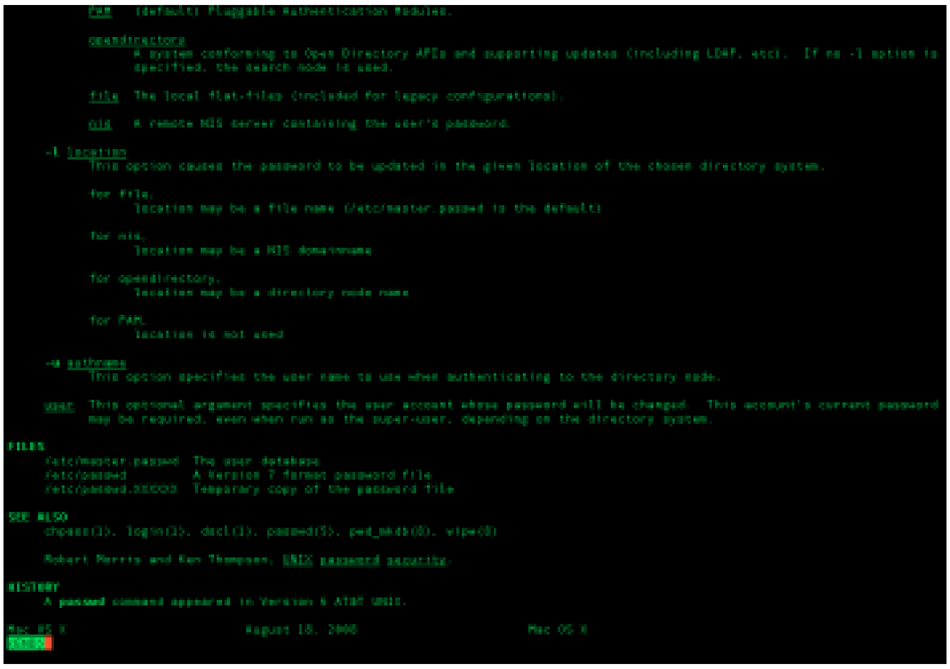
Password Detection

In a Linux environment

By running the command :

man -a passwd

in a terminal window, we can access comprehensive information about passwords. This includes details on how to manipulate the associated file to gather additional user information, the file's location, and other relevant data.



```
           PAM    (default) Pluggable Authentication Modules.

           opendirectory
                  A system conforming to Open Directory APIs and supporting updates (including LDAP, etc).  If no -l option is
                  specified, the search node is used.

           file   The local flat-files (included for legacy configurations).

           nis    A remote NIS server containing the user's password.

     -l location
           This option causes the password to be updated in the given location of the chosen directory system.

           for file,
                  location may be a file name (/etc/master.passwd is the default)

           for nis,
                  location may be a NIS domainname

           for opendirectory,
                  location may be a directory node name

           for PAM,
                  location is not used

     -u authname
           This option specifies the user name to use when authenticating to the directory node.

     user  This optional argument specifies the user account whose password will be changed.  This account's current password
           may be required, even when run as the super-user, depending on the directory system.

FILES
     /etc/master.passwd  The user database
     /etc/passwd         A Version 7 format password file
     /etc/passwd.XXXXXX  Temporary copy of the password file

SEE ALSO
     chpass(1), login(1), dscl(1), passwd(5), pwd_mkdb(8), vipw(8)

     Robert Morris and Ken Thompson, UNIX password security.

HISTORY
     A passwd command appeared in Version 6 AT&T UNIX.

Mac OS X                        August 18, 2008                        Mac OS X
(END)
```

Using the credentials provided by the "Information Technology Security" lab, we will gain access to the Linux environment hosted by SecLab. This will be done through a secure connection via the Secure Shell Protocol (SSH).

```
ssh asf_xxx@195.130.109.116 -p 9999
```

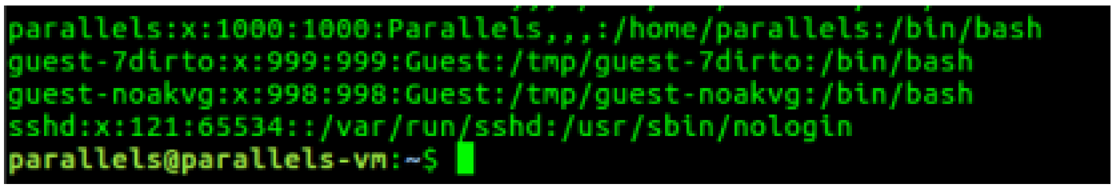
At this stage, the system prompts the user for a password. If the correct password is entered, access is granted; otherwise, the system denies entry.

By executing the command nano /etc/passwd, we can view the user records, where the first field represents the user, followed by the password field, which in this case is denoted by an 'x'.

Using the command ls -al /etc/passwd, we can examine the file's permissions:

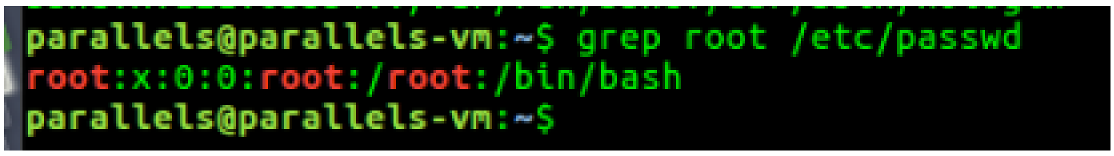
- **Root permissions**: -rw
  The root user has both read and write permissions for the file.
- **Group permissions**: --r
  The group has read-only permissions.
- **User permissions**: --r
  Each user also has read-only permissions.

Finally, running the command cat /etc/passwd provides a detailed view of the entries in the password file.



```
parallels:x:1000:1000:Parallels,,,:/home/parallels:/bin/bash
guest-7dirto:x:999:999:Guest:/tmp/guest-7dirto:/bin/bash
guest-noakvg:x:998:998:Guest:/tmp/guest-noakvg:/bin/bash
sshd:x:121:65534::/var/run/sshd:/usr/sbin/nologin
parallels@parallels-vm:~$
```

With the command: grep root /etc/passwd
We get the following results:

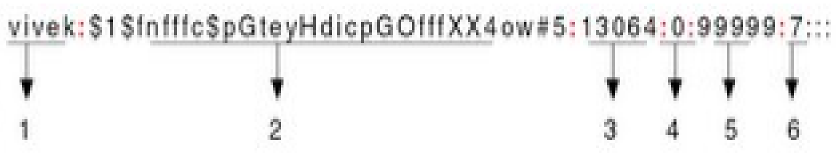


```
parallels@parallels-vm:~$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
parallels@parallels-vm:~$
```

The first field represents the username, which is used when the user logs into the system. Its length can range from 1 to 32 characters. The second field is the password field, where the character 'x' indicates that the password is encrypted. The encrypted password itself is stored in the shadow file, ensuring its security.

The third field represents the User ID (UID), which uniquely identifies a user within the system. Here, we see that the user root is assigned a UID of zero (0). UIDs from 1 to 99 are reserved for predefined system users, while UIDs ranging from 100 to 999 are allocated for administrative and system accounts or groups. The fourth field is the Group ID (GID), which corresponds to the user's primary group and is stored in the /etc/group file. The sixth field specifies the user's home directory, which, when logged in, is the default location the user will be placed in. This directory is represented by its absolute path. Finally, the seventh field indicates the shell the user will use, which is represented by the absolute path of the shell executable. In this case, the shell is /bin/bash.

Shadow file:



```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::
```

The fields in the shadow file are separated by colons (:). Subfields are separated by the $ sign.

- The first field is the user field.
- The second field is the password field. The password must be at least eight characters long, and a maximum of 12 characters long.

Typically, the format used by the shadow file for the hash field is defined as follows: $id$salt$hashed The first subfield in the password field indicates the algorithm used for encryption.

1. **$1$ is MD5**
2. **$2a$ is Blowfish**
3. **$2y$ is Blowfish**
4. **$5$ is SHA-256**
5. **$6$ is SHA-512**

- The second subfield is the salt. In cryptography, a salt refers to randomly generated data that is used as an additional input to a one-way hashing function. Its purpose is to enhance the security of the hash by ensuring that even if two identical inputs are hashed, their outputs will be distinct due to the unique salt applied to each.

- The third subfield is the Hashed Value, which takes the user's entered password and generates a uniquely encoded alphanumeric string using a one-way hashing function. Instead of comparing the actual password, the system compares the Hashed Value, which is produced each time the user logs in through the crypt function. This process ensures that the original password is never stored or directly compared.

The **`crypt`** function is an encryption tool that operates using a key as a filter, encrypting and decrypting data from stdin to stdout.

For example, consider a file named `simple.txt`, which contains a simple text in English.

By using the following command:

```
cat simple.txt | crypt > my.cpy
 Enter key :
```

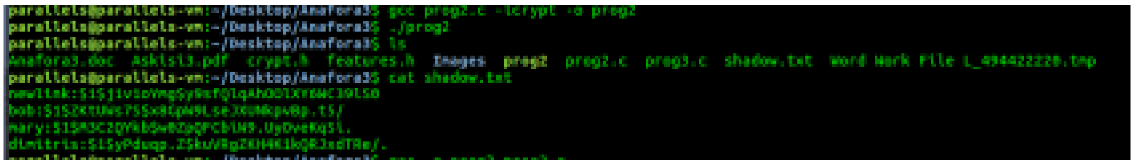
The system prompts for an encryption key, which is then used to transform the original text into an encrypted version. This encrypted text is directed through a pipeline and saved as my.cpy. If the password used for encryption is known, the original text can be decrypted and restored. To view the contents of the encrypted file, the following command is used: cat -n my.cpy.

In the passwd file, the third field indicates the date when the password was last changed.

- The fourth field represents the minimum number of days required before the password can be changed again.
- The fifth field specifies the maximum number of days before the password expires, at which point the user will be prompted to change it.

- The seventh field denotes the number of days after which the account will become inactive, effectively expiring the account.
- The eighth field outlines the absolute expiration date for the account, after which access is no longer permitted.

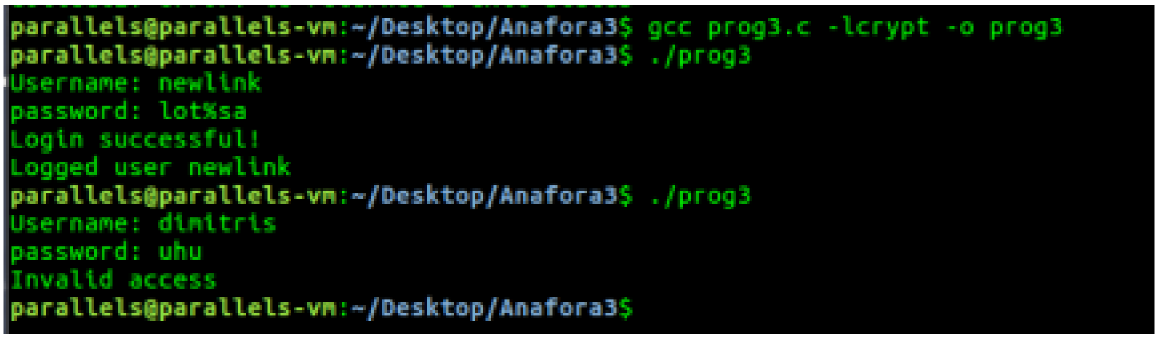
2) C language code is provided, followed by a screenshot showing the program execution.



```
parallels@parallels-vm:~/Desktop/Anafora3$ gcc prog2.c -lcrypt -o prog2
parallels@parallels-vm:~/Desktop/Anafora3$ ./prog2
parallels@parallels-vm:~/Desktop/Anafora3$ ls
Anafora3.doc  Asklsi3.pdf  crypt.h  features.h  Images  prog2  prog2.c  prog3.c  shadow.txt  Word Work File L_494422220.tmp
parallels@parallels-vm:~/Desktop/Anafora3$ cat shadow.txt
newlink:$1$jivioVmgSy6sf01qAhO01XY6WC39LS0
bob:$1$2KtUWs7S$x8GpN9LseJXUMkpvBp.tS/
mary:$1$M3C2QYkb$w8ZpQFCbiN9.UyOveKqS\.
dimitris:$1$yPduqp.Z$kuVRgZKH4K1kQR3xdTRe/.
```

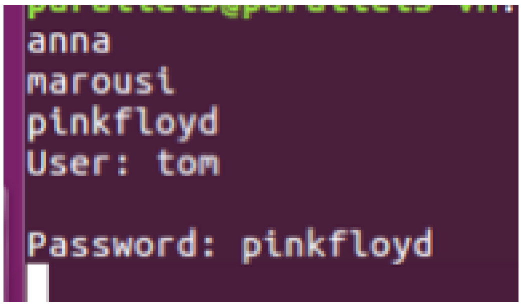
3) Also provided is the C code, accompanied by a screenshot illustrating the program's execution in two scenarios. In the first case, the user successfully logs into the system. In the second case, although the correct username is entered, the password is incorrect. As a result, the system displays a message indicating that the login attempt has failed.

It's important to note that, in the second case, the system uses a general error message to maintain security and avoid giving away specific details about the failure. This approach does not disclose whether the issue was with the username or the password. Instead, the system shows a generic message: *Invalid Access*.



```
parallels@parallels-vm:~/Desktop/Anafora3$ gcc prog3.c -lcrypt -o prog3
parallels@parallels-vm:~/Desktop/Anafora3$ ./prog3
Username: newlink
password: lot%sa
Login successful!
Logged user newlink
parallels@parallels-vm:~/Desktop/Anafora3$ ./prog3
Username: dimitris
password: uhu
Invalid access
parallels@parallels-vm:~/Desktop/Anafora3$
```

4) Note: because this specific question required the crypt function, the installation was done in a Pure Linux (Ubuntu) environment, so the display and screenshots are different from the previous questions. a) For the user tom, based on the information we have, we run the program with the passwords we created based on them.

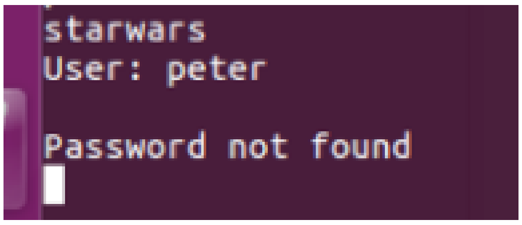


```
anna
marousi
pinkfloyd
User: tom

Password: pinkfloyd
```

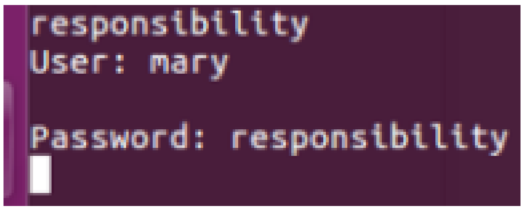
b) We enrich : dictionary.txt with more words and get the following

results :



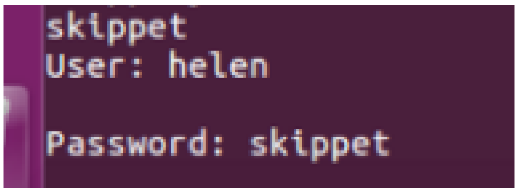
```
starwars
User: peter

Password not found
```



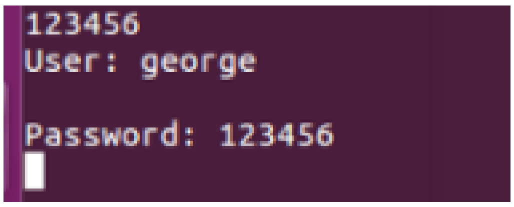
```
responsibility
User: mary

Password: responsibility
```



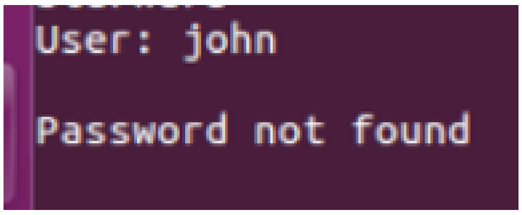
```
skippet
User: helen

Password: skippet
```



```
123456
User: george

Password: 123456
```

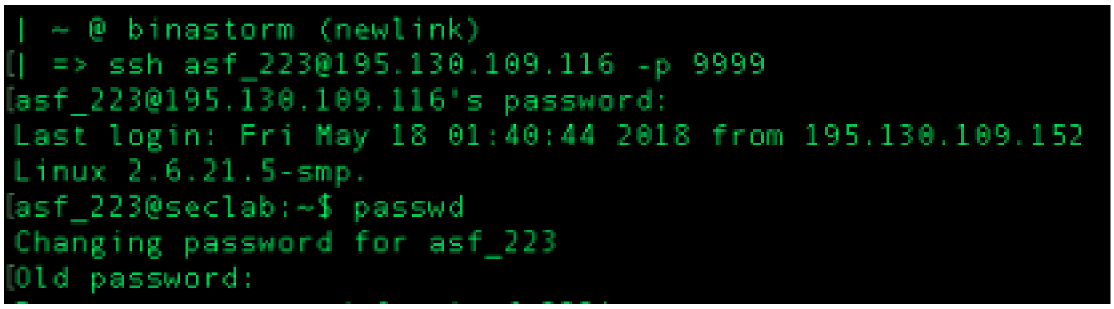


```
User: john

Password not found
```

c) We conducted a brute-force attack, testing all possible combinations. However, for users Peter and John, we were unable to uncover their passwords. This suggests that their passwords are likely not easily guessable or vulnerable to brute-force attacks.

5) With the credentials given to us we enter the seclab environment.



```
| ~ @ binastorm (newlink)
[| => ssh asf_223@195.130.109.116 -p 9999
[asf_223@195.130.109.116's password:
 Last login: Fri May 18 01:40:44 2018 from 195.130.109.152
 Linux 2.6.21.5-smp.
[asf_223@seclab:~$ passwd
 Changing password for asf_223
[Old password:
```

Where we have learned to **change the password to a more secure option**.

Sources used:

http://www.youssefkh.com/2014/01/13-free-ebooks-on-unix-and-linux.html
https://www.cyberciti.biz/faq/understanding-etcshadow-file/

Ambel Basha
Athens May 2018