

Introduction to Password Cracking with John the Ripper

Revision 1.3 for Raspberry Pi

Introduction

This laboratory exercise will provide some hands-on experience with password strength analysis using command-line tools in Linux. In this lab, the primary tool going to be used will be John the Ripper, a popular password cracking tool. This tool will be used to demonstrate the process of cracking weak and strong passwords using both dictionary and mask attacks.

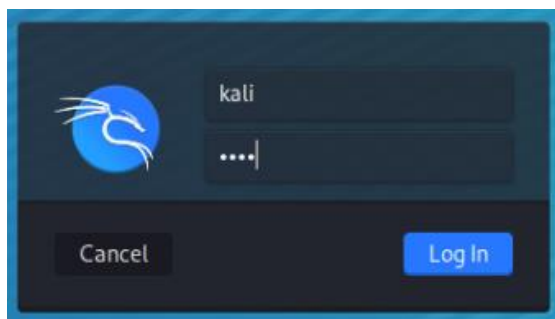
Objectives

Each student will be able to:

- Understand what makes weak and strong passwords
- Demonstrate creating hashes from plain text passwords
- Demonstrate how to use password cracking tools
- Understand the difference between dictionary and brute force password cracking

Setting up

1. Log onto your raspberry pi with the username “kali” and the password “kali”.



EXERCISE 1: INTRODUCTION TO PASSWORD AUDITING

On Linux systems, user accounts are stored in the **/etc/passwd** file (world-readable text file), and passwords are hashed and stored in **/etc/shadow** (a text file only readable by the root user). You have administrative (root) access on your Kali virtual machine – go ahead and "cat" those files to see what they look like.

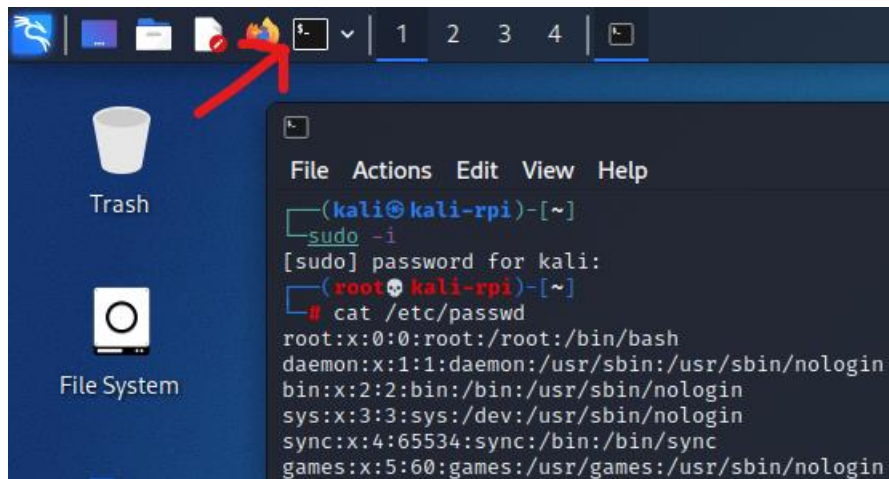
1. Launch QTerminal and give yourself administrative privileges by running:

```
$ sudo -i
```

Enter "kali" as the password when prompted. You need admin privileges to view the passwd and shadow files. Use the cat command to view the **/etc/passwd** and **/etc/shadow** files.

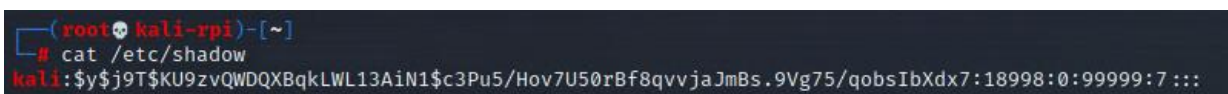
```
# cat /etc/passwd
```

```
# cat /etc/shadow
```



Take a look at the kali user (that's who you are logged on as) in the **/etc/shadow** file, and you will see the hashed password. You can scroll up and down using your mouse wheel.

Hint: You can use the scroll wheel on your mouse to look through the output in the terminal.



Note: For more information on the passwd file, you can explore <https://en.wikipedia.org/wiki/Passwd>. You'll notice that the hashed passwords are stored in the shadow file.

We'll use a password auditing tool called John the Ripper (JTR), one of the most effective and most widely known password crackers.

JTR can be run in various modes, including dictionary and hybrid modes, both of which use a "dictionary" (a list of passwords) provided by the user. Compiled from widely used passwords, these dictionaries contain words scraped from a company's website and other places. A good password dictionary listing is something that many cybersecurity professionals work to create over *years*!

EXERCISE 2: CREATE USERS WITH PASSWORDS

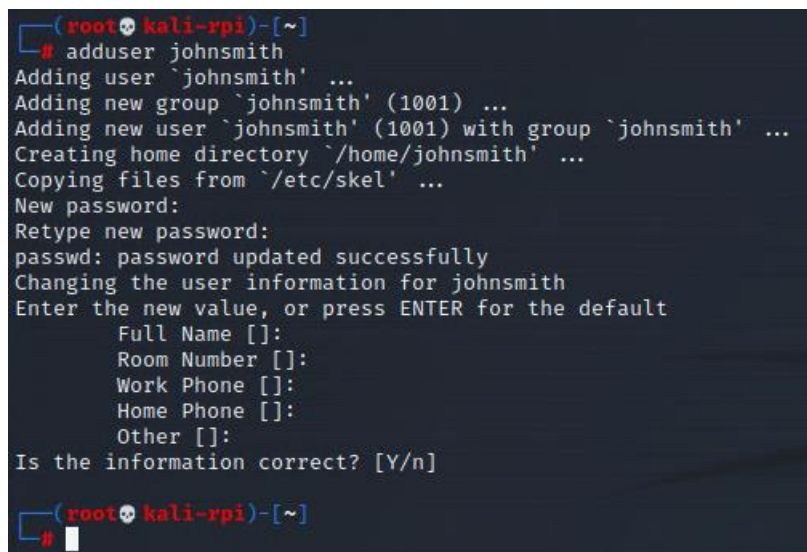
Here we will create a couple of new accounts on our Kali VM, one with a good password and one with a poor one:

2. Ensure you are in a 'root user' prompt (will be red-colored and ends with # instead of \$). If you are not, execute "**sudo -i**" and use "**kali**" for the password if asked/
3. Create 2 accounts as follows, one with a bad password and one with a good one.

Add a Linux user account for "johnsmith":

adduser johnsmith

When prompted, enter "**12345**" as the user password. The characters will be invisible when you type them into the terminal. Retype the password again to confirm. When asked for the user information (Full Name, Room Number, etc.), leave them blank.



```
(root@kali-rpi)-[~]
# adduser johnsmith
Adding user `johnsmith' ...
Adding new group `johnsmith' (1001) ...
Adding new user `johnsmith' (1001) with group `johnsmith' ...
Creating home directory `/home/johnsmith' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for johnsmith
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n]
Y
(root@kali-rpi)-[~]
#
```

EXERCISE 3: CRACK LINUX PASSWORDS

Now let's see which ones we can crack. Copy the **/etc/shadow** to a text file, and run JTR against it:

1. Change your current working directory to your desktop. This will make the subsequent steps easier.

cd /home/kali/Desktop

```
(root@kali-rpi)-[~]
# cd /home/kali/Desktop/

(root@kali-rpi)-[/home/kali/Desktop]
#
```

2. Copy the shadow file to your current directory

```
# cp /etc/shadow pass.txt
```

3. Use cat to verify the usernames and encrypted passwords.

```
# cat pass.txt
```

```
(root@kali-rpi)-[/home/kali/Desktop]
# cat pass.txt
root:*:18965:0:99999:7:::
daemon:*:18965:0:99999:7:::
bin:*:18965:0:99999:7:::
sys:*:18965:0:99999:7:::
sync:*:18965:0:99999:7:::
games:*:18965:0:99999:7:::
man:*:18965:0:99999:7:::
lp:*:18965:0:99999:7:::
mail:*:18965:0:99999:7:::
news:*:18965:0:99999:7:::
uucp:*:18965:0:99999:7:::
```

Scroll down and find the user you added, johnsmith, and note that they have a password hash next to their entries.

```
king-phisher:*:18965:0:99999:7:::
systemd-timesync:*:18965:0:99999:7:::
systemd-coredump:*:18933:0:99999:7:::
kali:$y$j9T$KU9zvQWDXBqkLWL13AiN1$c3Pu5/Hov7U50rBf8qvvaJmBs.9Vg75/qobsIbXdx7:18998:0:99999:7:::
xrdp:!:18998:0:99999:7:::
johnsmith:$y$j9T$ZNEG.IeLv7aPZiVQot3ti.$kl8bJ0xQtFZtIZ7742e32H/lQlNHQ0ZLzZaMzNF7z27:18998:0:99999:7:::
```

4. Let's start cracking! Use the following command to start cracking the password of the kali and johnsmith users

```
# john pass.txt --format=crypt
```

JTR will attempt to decipher the passwords and display any that it 'cracks' as it goes along. It starts in "single crack" mode, mangling username and other account information. It then moves on to a

dictionary attack using a default dictionary, then with a hybrid attack, then brute force. It will try every possible combination of characters (letters, numbers, and special characters) until it cracks them all.

On a typical computer, the password for the johnsmith account would be cracked quickly. However, a small raspberry pi will take a long time to crack even a basic password. Don't wait for John to crack johnsmith's password and instead proceed to the next step.

5. Press **[CTRL]+[C]** to stop execution. Exercise 4 will walk through using a wordlist to crack the johnsmith's simple password.

```
(root@kali-rpi)-[/home/kali/Desktop]
# john pass.txt --format=crypt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:05 10.60% 1/3 (ETA: 18:22:34) 0g/s 17.61p/s 17.61c/s 17.61C/s kali..kali999994
0g 0:00:00:10 12.71% 1/3 (ETA: 18:23:05) 0g/s 17.74p/s 17.74c/s 17.74C/s 999992..johnsmith99999q
Session aborted
```

A Deeper Look at JTR Files

John uses the following files to manage execution. Most are stored in the /usr/share/john folder on your Kali virtual machine (john.pot is stored elsewhere as indicated):

- **password.lst** *is john's default dictionary. You can specify another wordlist on the command line using the --wordlist= directive (for example # john --wordlist=/usr/share/dict/american-english /etc/shadow*
- **john.conf** *is read when JTR starts up and has rules for dictionary mangling for the hybrid crack attempt*
- **john.rec** *is used to record the status of the current password cracking attempt. If John crashes, it will start where it left off instead of starting again from the beginning of the dictionary.*
- **/root/.john/john.pot** *lists passwords that have already been cracked. If you rerun John on the same shadow file, it will not show these cracked passwords unless you delete this file first.*

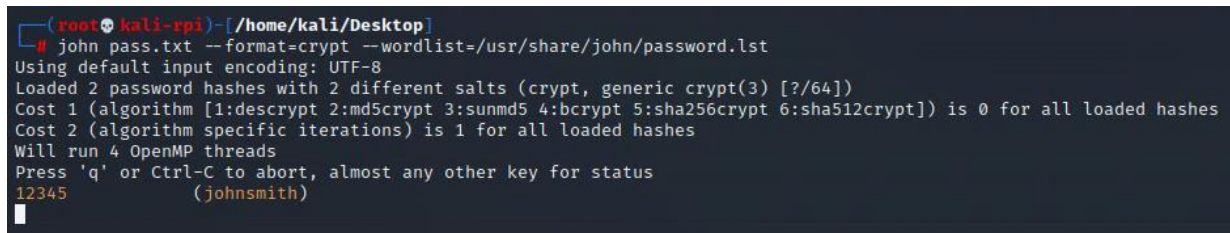
EXERCISE 4: MORE PASSWORD AUDITING

John the Ripper's default dictionary (described above) is a short list of common passwords. In this exercise we will use this dictionary to crack the johnsmith user's password

1. Run john with a wordlist by invoking the flag **--wordlist** directive at the command line.

```
# john pass.txt --format=crypt --wordlist=/usr/share/john/password.lst pass.txt
```

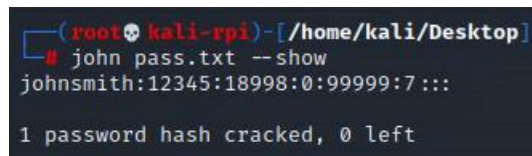
john will quickly run through the wordlist and crack the password of johnsmith



```
(root@kali-rpi)~/home/kali/Desktop
# john pass.txt --format=crypt --wordlist=/usr/share/john/password.lst
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
12345          (johnsmith)
```

2. You can view previously cracked passwords by using the **--show** flag. If john is still running, press **[CTRL]+[C]** to abort and use the following command to view the cracked password

```
# john pass.txt
```



```
(root@kali-rpi)~/home/kali/Desktop
# john pass.txt --show
johnsmith:12345:18998:0:99999:7:::

1 password hash cracked, 0 left
```

EXERCISE 5: MASK ATTACKS

Brute force attacks are where every possible combination of characters is combined and tried against a hash. The more intelligent form of this attack is called a mask attack. Instead of blindly attempting every possible combination, a mask attack uses known information to reduce the number of combinations, significantly speeding up the cracking process. For example, it is common for passwords to be in a word+digit format, like "pass123." A mask attack could capitalize on this known pattern and only attempt all 4 letters + 3 digits combinations instead of every 7 letter+symbol+digit combinations.

3. Start by creating a new password using the **mkpasswd** utility.

```
# mkpasswd -m sha512crypt | tee -a password.txt
```

Enter "UMGC-4786" as the password.


```
(root@kali-rpi)~[/home/kali/Desktop]
# mkpasswd -m sha512crypt | tee -a password.txt
Password:
$6$Qmd2BS9LMMYGQzzZ$8WvHAFF/V/h62oGkEcvRhp0pi6Pc5lFT95TCSSGDHsJVpI3c0UIBRBicRBXGeqe0TDa6/JkxIuwu0go1SLcdB.
```

4. Use **cat** on the password.txt file to view the password created with the SHA-512 hashing algorithm.

cat password.txt

```
(root@kali-rpi)~[/home/kali/Desktop]
# cat password.txt
$6$Qmd2BS9LMMYGQzzZ$8WvHAFF/V/h62oGkEcvRhp0pi6Pc5lFT95TCSSGDHsJVpI3c0UIBRBicRBXGeqe0TDa6/JkxIuwu0go1SLcdB.
```

Let's pretend we know that the password begins with "UMGC-" and is followed by four unknown digits. The password will not be in any standard dictionary. However, we know that the password is in a predictable format, giving us a loophole to exploit.

A simple method to capitalize on this predictability would be to use a tool like **crunch** to generate a custom dictionary with every combination in the format "UMGC-XXXX." However, the more efficient method would be to use a mask attack, which simultaneously generates and tests the dictionary against the password.

A mask is composed of known and unknown elements to describe what the password should look like. There are predefined "placeholders" used when creating a mask to represent different sets of characters. For instance, a "?u" represents any single upper-case ASCII letter, or a "?d" is every possible single digit. Here is a list of the most common ones:

- ?l = lower-case ASCII letters
- ?u = upper-case ASCII letters
- ?d = digits
- ?s = specials (all printable ASCII characters not in ?l, ?u or ?d)
- ?a = full 'printable' ASCII.

See <https://miloserdov.org/?p=5031#33> for an excellent introduction to all the possible mask placeholders.

5. As we already know the format of our password, we can use the static characters "UMGC-" plus 4 digit placeholders to define "UMGC-?d?d?d?d" as our mask.

Use the "--mask=" option in John to define the mask it should use.

```
# john --mask="UMGC-?d?d?d?d" password.txt
```

```
(root@kali-rpi)~[/home/kali/Desktop]
# john --mask="UMGC-?d?d?d?d" password.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 ASIMD 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
UMGC-4786 (?)
lg 0:00:00:35 DONE (2022-01-06 18:46) 0.02792g/s 243.0p/s 243.0c/s 243.0C/s UMG-6996..UMGC-3146
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

6. Like in Exercise 4, we can use --show option to see our cracked password hash.

```
# john --show password.txt
```

```
(root@kali-rpi)~[/home/kali/Desktop]
# john --show password.txt
?:UMGC-4786

1 password hash cracked, 0 left
```

REFLECTION QUESTIONS

1. A HASH IS A FUNCTION THAT CONVERTS ON VALUE TO ANOTHER. IT IS USED IN CRYPTOGRAPHY, COMPRESSION, DATA INDEXING, DATA INTEGRITY AND MORE. YOU CREATED PASSWORD HASHES WHEN GENERATING PASSWORDS FOR NEW USERS AND THROUGH THE **MKPASSWD** COMMAND. EXPLAIN THE SIGNIFICANCE OF HASHING AS IT RELATES TO THE LAB YOU JUST COMPLETED.
2. IF YOU WERE PERFORMING THIS AUDIT ON A DEVICE ON BEHALF OF YOUR EMPLOYER, WHAT WOULD THE ETHICAL IMPLICATIONS BE FOR THE PASSWORDS THAT YOU WERE ABLE TO CRACK?