# Linux User Authentication[1]

**Due Monday, March 21st, 2022 @ 11:59pm**

(**8%** of the total course grade)

## 1. Objective

In most contexts related to computer security, user authentication is the fundamental building block and the primary line of defense. In this lab, you will gain a better understanding of a popular user authentication method, Username and Password authentication, and will become familiar with common tools used for offline password attacks. This will help students understand how computer system users prove their identity to the system. Specifically, this lab will demonstrate how Linux implements the Username and Password authentication method. Note that the SSH protocol, also known as Secure Socket Shell, is a widely used network protocol that provides a secure access to a remote computer, supports many authentication methods and most importantly, the public key authentication for interactive and automated connections. In Linux, the username and password authentication mechanism authenticates users with their username and password credentials that are stored and protected in the /etc/shadow file. The /etc/shadow file is accessible only by a privileged user.

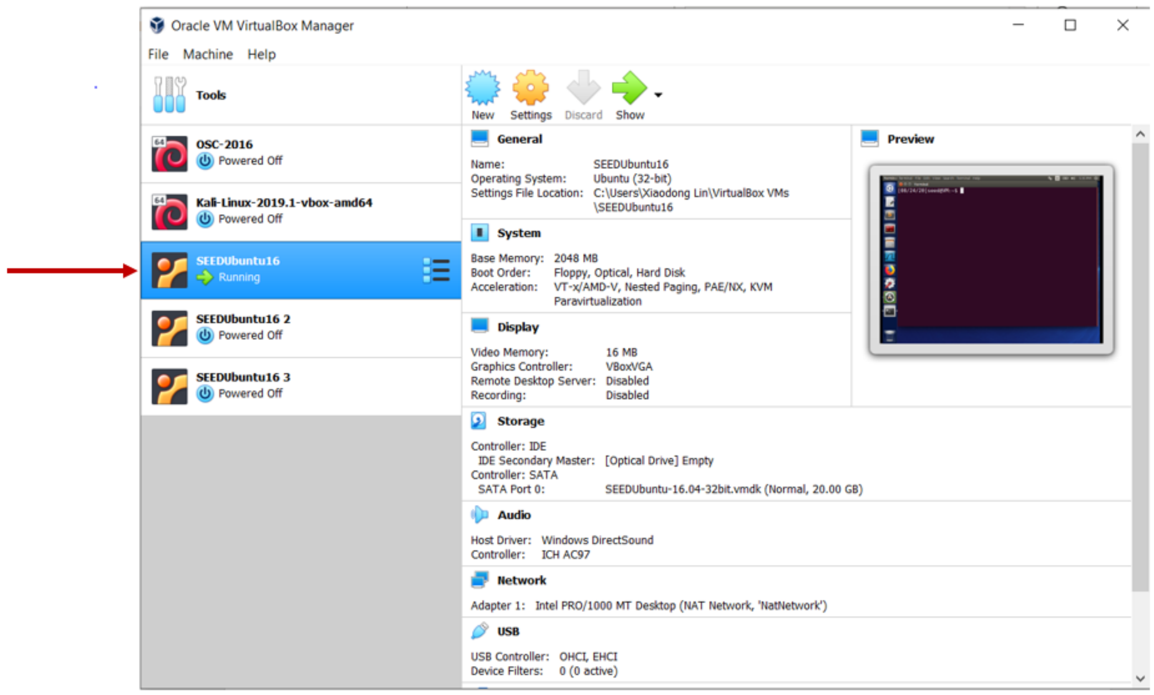
This lab will be graded. It has to be completed INDIVIUALLY, but you may want to discuss the lab content with your fellow students.

## 2. Environment Setup

IMPORTANT If you haven't set up your virtual Cyber Security Lab environment using VirtualBox on your laptop, please install and setup VirtualBox and Ubuntu 16.04 virtual machines by following instructions provided in **Lab 1**.

**Please note, for Lab 2, you only need to use one Ubuntu 16.04 VM.**

[1] Copyright © 2020 Xiaodong Lin, University of Guelph, Canada.
This lab may not be redistributed or used without written permission.



OSC-2016 — Powered Off
Kali-Linux-2019.1-vbox-amd64 — Powered Off
SEEDUbuntu16 — Running
SEEDUbuntu16 2 — Powered Off
SEEDUbuntu16 3 — Powered Off

**General**

```
Name:                    SEEDUbuntu16
Operating System:        Ubuntu (32-bit)
Settings File Location:  C:\Users\Xiaodong Lin\VirtualBox VMs
                         \SEEDUbuntu16
```

**System**

```
Base Memory:   2048 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, PAE/NX, KVM
               Paravirtualization
```

**Display**

```
Video Memory:           16 MB
Graphics Controller:    VBoxVGA
Remote Desktop Server:  Disabled
Recording:              Disabled
```

**Storage**

```
Controller: IDE
  IDE Secondary Master:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:           SEEDUbuntu-16.04-32bit.vmdk (Normal, 20.00 GB)
```

**Audio**

```
Host Driver:  Windows DirectSound
Controller:   ICH AC97
```

**Network**

```
Adapter 1:  Intel PRO/1000 MT Desktop (NAT Network, 'NatNetwork')
```

**USB**

```
USB Controller:  OHCI, EHCI
Device Filters:  0 (0 active)
```

Figure 1. Required Lab Environment Settings

1) Install the **John the Ripper** -- a fast password cracker onto one Ubuntu 16.04 virtual machine. To install **John the Ripper,** you need to be logged in as root or a user with sudo access. For example,

```
sudo apt-get install john
```

## 3. Linux User Authentication

**Authentication** is the way that a **user** is identified and verified by a system. In Linux, a password is a secret that is chosen by a user and shared between the user and a system. For security reason, Linux uses two separate files for user names and passwords. Furthermore, the passwords are hashed with salts where the password salts are random numbers chosen for each user by the system. Also, the /etc/**shadow file** is readable only by the **root-privileged user** and is therefore less of a **security** risk because they aren't available to regular users.

### /etc/passwd

The /etc/passwd stores user names. It contains one entry per line for each user (or user account) of the system. Each line is a string with 7 fields separated by a colon (:) symbol. A typical line looks like the following

```
student:x:1000:1000:student:/home/student:/bin/bash
```

The 7 different fields are:

1. Username
2. Password: An x character indicates that protected password is stored in /etc/shadow file, which be detailed later.
3. User ID (UID)
4. Group ID (GID)
5. User ID Info
6. Home directory
7. Command/shell

Note that if you check the permission of the /etc/passwd file by the following command, you can find out that the owner of the the /etc/passwd file is root but can be accessible to all users.

**/etc/shadow**

The /etc/shadow file stores user passwords as hashes in a particular format. It contains one entry per line for each user. Each line is a string with 9 fields separated by a colon (:) symbol. A typical line looks like the following

```
student:$6$hNYeQrax.8/7gwno$P4sMuDUAvX3r/seBcnOd7VxHtVcomq9ASMPhv7SRc.XGWocNIftaV
BTRde4v4fCVlaibSBNJ9cXdxqRtJhIGP1:15713:0:99999:7:::
```

We are particularly interested in the first two fields, which are:

1. The user name
2. The salted password hash

Let’s take a close look at the above example. It can be observed that the hash field itself is comprised of three different fields. They are separated by '$' and represent:

1. Some characters which represents the cryptographic hashing mechanism used to generate the actual hash
2. A randomly generated salt to safeguard against rainbow table attacks. Rainbowtables are nice attack vector against hashes. The idea behind this concept, is to simply pre calculate all possible hashes and then just lookup a hash in the tables to find the corresponding password.

3. The hash which results from joining the users password with the stored salt and running it through the hashing algorithm specified in the first field. The details can be found below

So that first field before the salt and actual hash can have a finite set of possible values. The standard methods supported by GNU/Linux are:

| The first field | Algorithms |
|---|---|
| $1$ | md5 |
| $2a$ | Blowfish |
| $2y$ | Blowfish, with correct handling of 8 bit characters |
| $5$ | sha256 |
| $6$ | sha512 |

Given the password hash in the above example, if you wanted to check if a given password matched it you would run the following python code

```
import crypt
print crypt.crypt('password', '$6$TDaixhdcurTsYkcu$')
```

Figure 1: Sample Python Code for Salted Password Hashing

The above python code should print the hashed password "password" with the salt "TDaixhdcurTsYkcu" using sha512. You can then compare this hash with the hash in /etc/shadow file. If they match, it means that the password is correct and the user is legitimate

Note that if you are new to Python, an interpreted high-level programming language, the followings are some basics about python

- Run a python script from the python prompt from within the shell terminal

```
[root@localhost student]# python
Python 2.7.3 (default, Apr 30 2012, 21:18:10)
[GCC 4.7.0 20120416 (Red Hat 4.7.0-2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import crypt
>>> print crypt.crypt('password', '$6$TDaixhdcurTsYkcu$')
$6$TDaixhdcurTsYkcu$ttCvqIeY8iiHH70UihdUl0TWpQoTu07aVp0mDNn39
zG2qRe80UdoJ7YdHacRzFRnhKjLPxbWH88RQ0WZ230Dc/
>>> exit()
```

- Run a Python program (or python script) (assuming that saltedpasswd.py is a Python program)

```
[root@localhost student]# python saltedpasswd.py
This program calculates salted hashed password!
$6$TDaixhdcurTsYkcu$ttCvqIeY8iiHH7oUihdUl0TWpQoTu07aVp0mDNn39
zG2qRe8oUdoJ7YdHacRzFRnhKjLPxbWH88RQoWZ230Dc/
```

where saltedpasswd.py looks like

```
# This program calculates salted hashed password!
import crypt
print('This program calculates salted hashed password!')
print crypt.crypt('password', '$6$TDaixhdcurTsYkcu$')
```

There are many online resources on python. For a quick tutorial on python, please go to the following website

http://thepythonguru.com/

**Attacks against Linux User and Password Authentication**

Unfortunately, a system could be compromised and hacked. As a result, the /etc/shadow file could be available to the attacker. Even though a password is hashed with a salt, there are still some attack vectors against the password hashes. For example, the attacker could keep a dictionary of popular passwords and try them automatically. Actually, there are a lot of dictionaries available on the internet. Another approach would be to just try out all possible combinations of characters which will consume a huge amount of time. This is known as brute force attack.

There are many tools available on the internet that can execute dictionary or brute force attacks against user passwords of different operating systems. An example of them is **John the Ripper**. It is a fast password cracker, currently available for many flavors of Unix and Windows. Its primary purpose is to detect weak Unix passwords.

## 4. Lab Exercises

### 1) Password Cracking

Suppose you know a Linux system user with username “cis4510suser” uses one of ten passwords listed below

```
abc1
abc2
abc3
abc4
abc5
abc6
```

You certainly know that an attacker can brute force all 10 passwords and try them one by one on the system to find out the correct one. However, in the real world, it won't work well on a production system, since a user account could be locked due to too many failed password attempts, For example, to allow for user error and to thwart brute force attacks, **4** is a popular setting, which means that 4 failed log-in attempts will cause a user account to be locked.

Now we assume that you have compromised the Linux system where the user "cis4510suser" belongs to. Particularly, you are able to gain access to the /etc/shadow file. The line for the user "cis4510suser" is shown below

```
cis4510suser:$6$PZOyfkQz$C8Rj2dcT2kjTHKYDJDzCMJ7GpJeklx7pafQjyONMdMqcBAeqcW/tDfEB
OPczx7VdCF70E5CgXsG4krDZBjkmv0:17551:0:99999:7:::
```

**Q1:** What is your method to find the correct password of user "cis4510suser"? **Your method should successfully discover the correct password but without any need of guessing passwords to the system**. Note that this is because a modern system is usually protected against password guessing attacks through account lockouts. Account lockouts are used to lock out an user account if the system observes multiple failed attempts (e.g., 4) to log into the account. It means your attacks fails if you directly guess passwords to the system. (HINT: you can reuse the Python code snippet shown in Figure 1 to crack the password of the user "cis4510suser".) (3 marks)
**Note that in order to receive full credit for this question, you must: (1) submit your the Python program (named q1.py) you developed to determine the password used by user "cis4510suser" (2 marks), and (2) briefly explain how your program works (1 mark).**

**Q2:** What is the password for user "cis4510suser"? (1 mark)

## 2) Unix Password Hashes Cracking using John the Ripper

a) *Creating usernames and passwords*

- Create the following new usernames as well as and their corresponding passwords using the *useradd* and *passwd* commands. To add users using the useradd command, you need to be logged in as root or a user with sudo access. For example, the following command adds a new user named test1 to your system:

```
sudo useradd -m test1
```

where the "-m" option means the default home folder is created if it does not exist. Then, you can use the *passwd* command to change the password for user test

```
$ sudo passwd test1
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

| User | Password |
|---|---|
| user1 | password |
| user2 | Password |
| user3 | Passw0rd |
| user4 | Passw0rd1 |

**Note that for the rest of the lab exercises using John the Ripper, we use the default setting for it.**

b) *Cracking Passwords with John the Ripper*

After you installed John the Ripper, you are ready to use John the Ripper to crack passwords, for example,

```
john mypassword
```

where *mypassword* is a password file (or the traditional Unix password file). For example, /etc/shadow in the system used in our lab exercise is a password file.

In the next attack scenario, we assume that an attacker has gained access to the password file, /etc/shadow. So you need to make a copy of your shadow passwords file, for example,

```
cp /etc/shadow /home/seed/.
```

Note that you need to be logged in as root or a user with sudo access, for example,

```
sudo cp /etc/shadow /home/seed/.
```

IMPORTANT **Don't work on the /etc/shadow directly.**

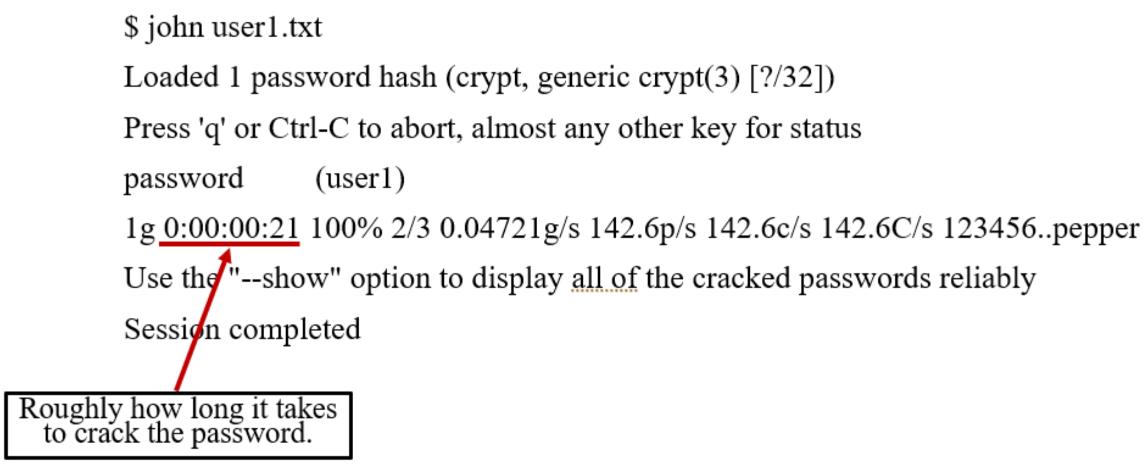
Afterwards, fetch the corresponding line for each user (user1 or user2 or user3 or user4) from the file /home/seed/shadow, and save it into four different files, each of them for one user. For example, you can generate a password file user1.txt for user user1, assuming that you make a copy of the file /etc/shadow in /home/seed

```
sudo grep user1 /home/seed/shadow > user1.txt
```

Then, you can use John the Ripper to crack the password of user user1, for example,

```
john user1.txt
```

The following is example output of John the Ripper successfully cracking one password



```
$ john user1.txt
Loaded 1 password hash (crypt, generic crypt(3) [?/32])
Press 'q' or Ctrl-C to abort, almost any other key for status
password        (user1)
1g 0:00:00:21 100% 2/3 0.04721g/s 142.6p/s 142.6c/s 142.6C/s 123456..pepper
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

From the above example, it takes roughly 21 seconds to crack the password.

Note that to display cracked passwords, use "john --show" on your password hash file(s).

Give it time to see how long it takes for each password to be cracked. Record those times here: Please note that

| User Names | Timing for cracking the password |
|---|---|
| user1 | Q3: ______________ (0.5 mark) |
| user2 | Q4: ______________ (0.5 mark) |
| user3 | Q5: ______________ (0.5 mark) |
| User4 | Q6: ______________ (0.5 mark) |

Note that if a password hasn't been cracked after 5 minutes, you simply force John the Ripper to exit by "Ctrl+c" and record the time as "Not cracked".

**Reflection Questions:**

**Q7**: In your opinion, which password is the most complex one, password, Password, Passw0rd or Password1? Did you notice a correlation between the times it took to crack a password versus the complexity of the password? You should have seen that more complex passwords take longer to recover. (1 mark)

______________________________________________________________________

______________________________________________________________________

**Q8**: Passwords are a commonly-used method of authentication. One weakness with password systems is the choice of the password. Studies show that 81% of data breaches worldwide are caused by hacked passwords. It can also be observed in the above exercises that you need to choose your password carefully and avoid using weak passwords. A weak password is one that can be easily guessed or broken, for example, “password”, “Password”. Unfortunately, the tendency to select weak passwords has led to a number of system break-ins, some quite highly publicized: [4]. Conduct a study by surveying some online articles and **give one tip to choose a strong password**. (1 mark)

______________________________________________________________________

IMPORTANT From what we have done in the above, you know you need to choose a strong password to protect your account.

c) *Using John the ripper, try to break the passwords in the passwords.txt file provided.*

**Q9:** What passwords were you able to crack? (2.5 marks)

______________________________________________________________________

**Note that you should abort John the Ripper after it has been running for 5 minutes. In other words, any remaining password is considered not cracked. To display cracked passwords, use “john –show” on your password hash file(s).**

## 3) Cracking the password in Q1 using John the Ripper

Recall from Q1, Section 4.1 that the line for the user “cis4510suser”, which is shown below

```
cis4510suser:$6$PZOyfkQz$C8Rj2dcT2kjTHKYDJDzCMJ7GpJeklx7pafQjyONMdMqcBAeqcW/tDfEB
OPczx7VdCF70E5CgXsG4krDZBjkmv0:17551:0:99999:7:::
```

**Q10:** Clearly describe what you do to use John the Ripper to crack the password of user “cis4510suser” (1 mark)

**Q11:** Does John the Ripper successfully crack the password for user “cis4510suser”? (Yes/No) (0.5 mark)

**Note that you should abort John the Ripper after it has been running for 5 minutes.**

### 4) Bonus exercises

**Please note that you are not required to complete these bonus exercises but you are encouraged to try them. Doing so can yield you some bonus points for the final grade of the course.**

**BQ1:** Suppose you are now able to obtain the line for the user “mctisuser” in the /etc/shadow file on a Linux system, which is shown below

```
mctiuser:
$6$wDRrWCQz$QaU0lsixlFkIojWVbSPwkn.ZRex1f.H089auCQjEsnZu0Nnm.HV/yCQ74sxYU
2yGBO75Y0uZVZN6Lnn/RxQUV1:17372:0:99999:7:::
```

Now you only know that the length of the password used by the user “mctiuser” is 4 letters (all lowercase). What is the password for user “mctisuser”? (2 marks)

**Note that in order to receive full credit for this question, you must: (1) submit your the Python program (named bq1.py) you developed to determine the password used by user “mctisuser” (0.5 mark), (2) briefly explain how your program works (0.5 mark), and determine the password used by the user “mctiuser” (1 mark).**

Now you should be able to design and develop your own password cracker, which could be better than John the Ripper.

## 5. Submission

You can submit your answers in one zip file (containing your developed python programs and a text file named lab2_ FirstName_FamilyName.txt with your answers to the questions in the lab).

The zip filename must be lab2_FirstName_FamilyName.zip, where FirstName is your given name and FamilyName is your family name in Pinyin. This naming convention facilitates the tasks of marking for the instructor and course TA. It also helps you in organizing your course work. Failure to follow the requirements will result in mark reduction.

## Reference:

[1] John the Ripper password cracker. http://www.openwall.com/john/
[2] https://www.aychedee.com/2012/03/14/etc_shadow-password-hash-formats/
[3] https://unix.stackexchange.com/questions/2366/program-for-decrypt-linux-shadow-file
[4] Ransomware attacks: Weak passwords are now your biggest risk. https://www.zdnet.com/article/ransomware-attacks-weak-passwords-are-now-your-biggest-risk/