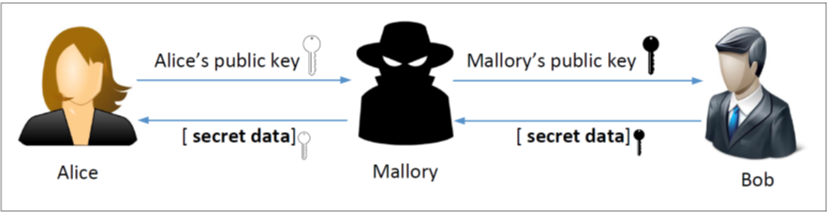
Introduction

(Estimated time 3-6 hours) - Symmetric encryption algorithms are fast but they are weak due to the fact that the sender and receiver need to share the same key. Public Key Cryptography such as Asymmetric encryption solves the shared key problem but is slow. The Diffie-Hellman key exchange algorithm allows us to securely exchange a shared key over public networks to allow for the secure exchange of symmetric keys. This approach is in use today when we communicate with websites where an encrypted channel for confidentiality is a concern.

One of the problems we encounter is the "Man in the Middle" (MITM) attack whereby an attacker can replace the intended recipients public key with the attackers. This could lead Alice, in this case to be sending confidential information to Mallory (Attacker) who has now spoofed Bob's address.



MITM attacks are defeated through Digital Certificates that are signed by a trusted authority and linked to an entity like example.com or Bob.



Getting started heading and video atom:

Learning Outcomes

In the following Lab exercises we will work with OpenSSL to create certificates for a student created website.

**Note:** It is Illegal to try these exercises outside of our Lab Environment.

Lab Assessment information (with screenshots) atom:

## Objectives

* Learn about certificate generation.
* Know the function of certificate authorities.
* Learn about self-signed certificates.

Help and Support

For more information on using Practice Labs, please see our **Help and Support** page. You can also raise a technical support ticket from this page.

Click Next to proceed.

Copyright atom:

Lab Topology

During your session, you will have access to the following lab configuration.

**Practice Labs to Add Image**

Depending on the exercises you may or may not use all of the devices, but they are shown here in the layout to get an overall understanding of the topology of the lab.

* **PLABSEED** – (SEED version of Linux) - 192.168.0.3

Exercise 1 – Becoming a Certificate Authority

A Certificate Authority (CA) is a trusted entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. A number of commercial CAs are treated as root CAs; VeriSign is the largest CA at the time of writing. Users who want to get digital certificates issued by the commercial CAs need to pay those CAs.

In this lab, we need to create digital certificates, but we are not going to pay any commercial CA. We will become a root CA ourselves, and then use this CA to issue certificate for others (e.g. servers). In this task, we will make ourselves a root CA, and generate a certificate for this CA. Unlike other certificates, which are usually signed by another CA, the root CA’s certificates are self-signed. Root CA’s certificates are usually pre-loaded into most operating systems, web browsers, and other software that rely on PKI. Root CA’s certificates are unconditionally trusted.

### Task 1

Power on your SEED Linux server. For this lab, you will clone a github repository holding the instructions to the lab for easier copy and paste functionality.

Perform the following commands:

**cd Desktop/**

**sudo git clone** [**https://github.com/remani/SRA221-PSU-Hozza.git**](https://github.com/remani/SRA221-PSU-Hozza.git)

There should now be a file on your desktop. Double click on it and go to the PKILab subfolder. You should now double click on PLABS Instructions PKI Lab.

### Task 2

In order to use OpenSSL to create certificates, you have to have a configuration file. The configuration file usually has an extension .cnf. It is used by three OpenSSL commands: ca, req and x509. The manual page of openssl.conf can be found using Google search.

Perform the following commands:

**cp /usr/lib/ssl/openssl.cnf ./**

**mkdir demoCA**

**cd demoCA**

**mkdir certs crl newcerts**

**touch index.txt serial**

**echo 1000 > serial**

**cd ..**

As we described before, we need to generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. You can run the following command to generate the self-signed certificate for the CA:

**openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf**

You will be prompted for information and a password. Do not lose this password, because you will have to type the passphrase each time you want to use this CA to sign certificates for others. You will also be asked to fill in some information, such as the Country Name, Common Name, etc. The results of the command are stored in two files: ca.key and ca.crt. The file ca.key contains the CA’s private key, while ca.crt contains the public-key certificate.

**Take a Screenshot** output of your new CA private Key. Your output should be a cat or more of the key file. We only need to see the top portion of the output (Lines 1- 10 are fine).

**Take a Screenshot** output of the Public Key Cert. Your output should be a cat or more of the Public Key Cert file. We only need to see the top portion of the output (Lines 1- 10 are fine).

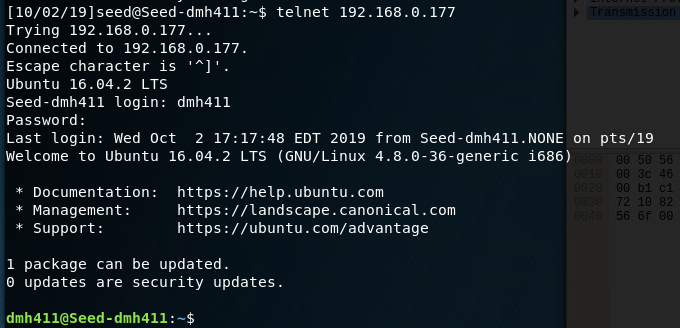
### Task 2

On the Kali machine, you should telnet to the SEED machine.

Open up a terminal prompt and enter

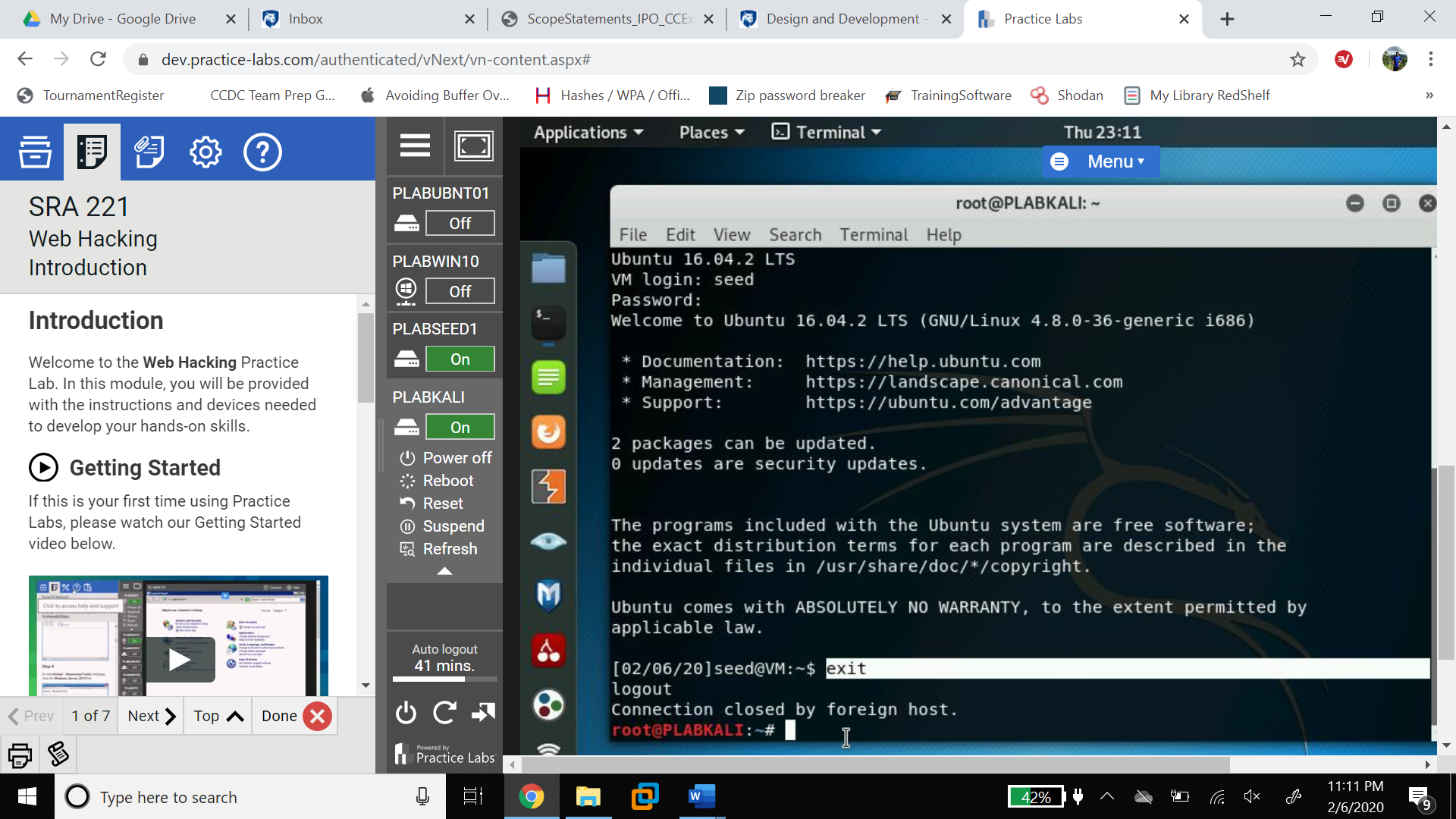
**telnet 192.168.0.3**

You should receive a prompt to enter a login. Enter your psu id that you added in the previous task and hit enter. Then it should prompt you for the password. Enter the password that you set for the new user and hit enter…. You are now in the SEED server, see example below



**Take a Screenshot** of your successful telnet session to the seed server from Kali.

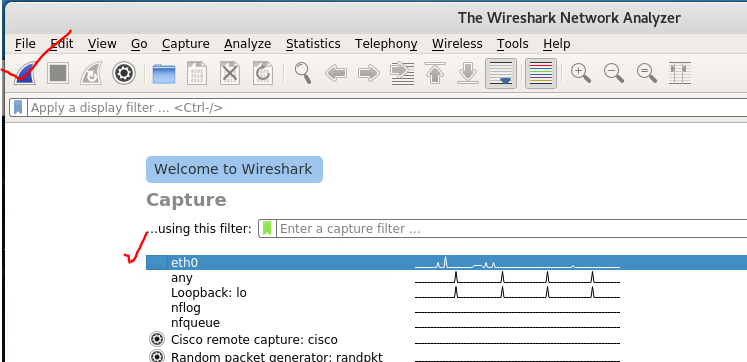
### Task 3



Type **exit** in the terminal to go back to the Kali Server.

Now we will sniff the traffic and find the unencrypted password using Wireshark. Open another terminal window on the Kali server (right click on the Kali Desktop) enter **sudo wireshark**, this will start wireshark as root user. Ignore any LUA errors.

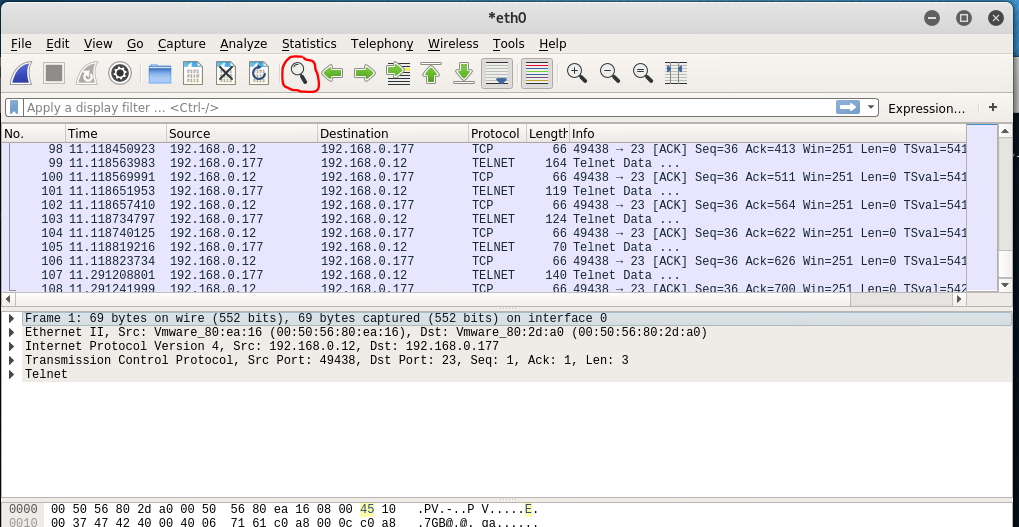
Wireshark will capture the network packets as they go across the network and allow us to try and grab the password. Start a capture by clicking the Blue Shark Fin in the upper left to start a capture. Make sure eth0 is highlighted, as this is your network interface.



Once the capture starts go back and telnet to the SEED machine again just as you did in task 2. IN the meantime, you should see packets capturing in your Wireshark Session.

STOP the Wireshark capture by clicking on the RED box in the upper Left.

SAVE the capture in case we want to look at it later, in the upper left, under file do a save as and save it as whatever name you prefer to your desktop or other folder.



Now to find the password you want to right click on the first Telnet Data Packet and select "follow" the “TCP stream.” Your username login will be echoed as 2x but the password should be visible as plain text.

**Take a screenshot** to show your username and password.

***\*\*\*\*\* Remember to go back to Practice Labs occasionally and click the auto logout to extend your session and not get automatically kicked out \*\*\*\*\****

Exercise 2 – Apply Encryption

Now we will try to login with an encrypted protocol called ssh or (Secure Shell). We can capture these packets as well to prove that our password and data is confidential.

### Task 1

Our goal in this task is to login again from Kali to SEED but in a more secure fashion. Start a Wireshark Session to capture the SSH traffic.

From the Kali terminal, type the following to login securely to SEED

**ssh –l {yourid} 192.168.0.3**

The first time we do this the server complains about the authenticity of the server and gives us a message.

**Take a Screenshot** showing the message

**Question**

What is the significance of the SHA-256 value we see?

* The Kali Machine’s ECDSA public key
* The Seed Machine’s ECDSA public key (Correct)
* The Kali Machine’s ECDSA private key
* The Seed Machine’s ESDSA private key’

Type “**yes**” and hit enter and then type the password you set and hit enter. You should be logged in. Then type “**exit**” to exit the remote shell.

STOP your Wireshark session but remember to save it as you will analyze it in task 2.

Find the known\_hosts file on the Kali machine and **Take a Screenshot** of it (You may have to research where/what it is)

### Task 2

**Take a Screenshot** of the Wireshark capture that shows the protocol “SSHv2”

(If you do not see any SSHv2 packets then you might have to delete the known\_hosts file to initiate a new exchange).

“**SSHv2: Server: Key Exchange Init**” is contained in the header of a packet that should be on your trace. This is where the client and server decide which algorithms they support and decide what to use. They also negotiate compression, hashing and other settings.

**Take a Screenshot** of the Follow TCP Stream on the Server Exchange Init. (Remember this is done by right clicking on the packet and going to “follow”).

**Question**

(True or False) The client also does a key exchange init

* True (Correct)
* False