Encryption
Cryptography with OpenSSL

Modified by: Matt Kirkland, Jonathan Buch, & Ananth Jillepalli
Based on: Crypto Lab: PublicKey Cryptography and PKI, by Wenliang Du

August 17, 2017
Version 1.0

University of Idaho

CS 439: Applied Security Concepts

Abstract

This tutorial will give students practical experience in encryption services, using the OpenSSL library. Public Key Infrastructure and various flavors of symmetric encryption will be covered. This work is built upon Dr. Du Wenliang's document: Crytpto Lab: PublicKey Cryptography and PKI.

Contents

1 Objectives of this Tutorial >

Students will...

1. Be introduced to OpenSSL and review encryption concepts;
2. Experience with OpenSSL for PKI & symmetric encryption;
3. Understand trust between a client and server using a CA.

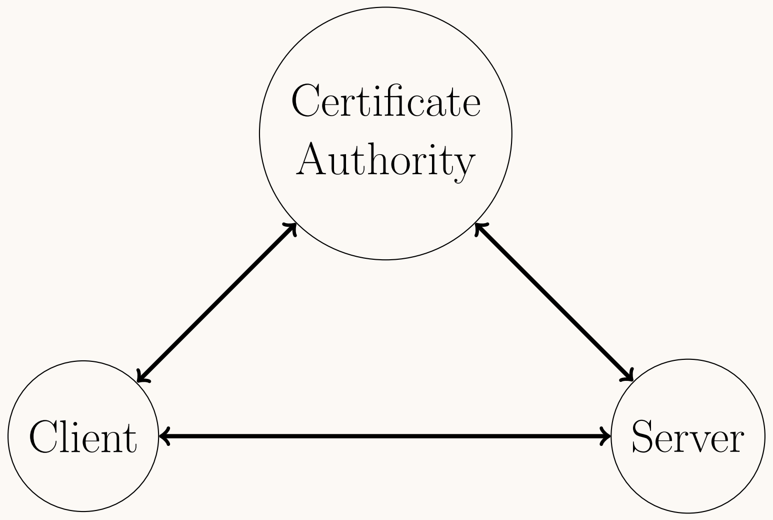
2 Required Background <>

1. Basic understanding of encryption concepts (Symmetric vs Asymmetric);
2. Ability to work with virtual machines (i.e. VirtualBox, VMware, etc.);
3. Basic understanding of security concepts.

3 Hardware & Software Requirements <>

1. A computer that can smoothly run 2 separate Virtual Machines (VMs) simultaneously;
2. A virtualization platform like VMware or VirtualBox;
3. A Seed Ubuntu 12.04 VM & a Seed Ubuntu 12.04 server VM.

4 Public Key Infrastructure



1. PKI - ”the distribution and identification of public encryption keys, enabling users and computers to both securely exchange data over networks such as the Internet and verify the identity of the other party.” [2];
2. Use widely in the Internet as the main means of authenticating different entities to each other. [2];
3. The PKI system relies entirely on trust and the CA. [2].

5 Related News <>

1. "Regulators Warn of Man-in-the-Middle Attack Risks" - ICS CERT recommendations [1];
2. "Encrypted Chat" - Wired [3].

1. The expected increase of man in the middle attacks has financial regulators becoming increasingly concerned. As a result, ICS-CERT has released a series of measures that they believe will help. Among these measures is updating security protocols like TLS and SSL as well as utilizing certificate pinning. Certificate Pinning forces a certificate to be trusted solely or allow it to trust other certificates signed by it [1].

2. Messaging apps are becoming more popular by the day. With this increase, there is now an increased demand for encrypted phone calls. Unfortunately, there are many challenges to this process; one of them being: reliability. Many advances come from companies like Skype, Google, and Apple. A major contender in this field is currently Cryptophone. However, their solution is not just an app but separate hardware baking into a new cellphone. CryptoPhones encrypt using AES and Twofish. [3].

6 Scenario: Man in the Middle Attack <>

You are an employee in charge of the new online payment system for the company website. What are the risks to be considered with the handling of customer personal and payment information? How might you address these concerns?

Time Required: 15 Minutes

7 Questions 1,2 <>

Q1: What is Public Key Infrastructure?

Q2: How is the trust between a client and server established using PKI?

Answer to Q1:
Public Key Infrastructure involves using a CA and certificates to authenticate difference entities. Typically this is a web-server and a client.

Answer to Q2:
The client checks the server certificate and trusts the certificate based on the CA that issued it. If it trusts the CA, it will trust the certificate and thus the server.

8 Task 1: Compiling OpenSSL Library <>

On the Server VM and Client VM:

1. `cd /home/seed/openssl-1.0.1/`;
2. `sudo ./config`;
3. `sudo make`;
4. `sudo make test`;
5. `sudo make install`.

Estimated Time: 5 Minutes;
Deliverable: The compiled OpenSSL Library.

9 Task 2: Create the Certificate Authority <>

On the Server VM:

1. Create the neccessary directories;
2. Create the necessary files;
3. Create the CA and record it's information.

Estimated Time: 25 Minutes;
Deliverable: A self-signed Certificate Authority.

On the server VM:

1. Create the necessary directories
   - (a) `cd /home/seed/openssl-1.0.1/`
   - (b) `sudo mkdir working`
   - (c) `cd working`
   - (d) `sudo mkdir demoCA`
   - (e) `cd demoCA`
   - (f) `sudo mkdir certs`
     `sudo mkdir newcerts`
2. Prepare the necessary files
   - (a) `sudo gedit index.txt`
     Then, save the blank file and exit gedit.
   - (b) `sudo gedit serial`
     Insert an even number of digits into this file, save, and exit.
3. Create the CA
   - (a) `cd ..`

(b) `sudo openssl req -new -x509 -keyout ca.key -out ca.crt -config /usr/lib/ssl/openssl.cnf`
This creates the CA's key and uses it to sign the CA's certificate.

(c) Please copy down the information so that way it can be referenced later. A document has been provided for you to enter this information into, for your own convenience.

10 Task 3: Client Setup <>

1. Create key pairs for the client;
2. Create the client CSR;
3. Create the client certificate.

Estimated Time: 25 Minutes;
Deliverable: A signed client certificate.

On the Client VM:

1.
   ```
   sudo mkdir working
   cd working
   ```

2.
   ```
   sudo openssl genrsa -aes128 -out client.key 1024
   ```
   Record client key information.

3.
   ```
   sudo openssl req -new -key client.key -out client.csr
   -config/usr/lib/ssl/openssl.cnf
   ```
   Record client certificate information.

On the Server VM:

1.
   ```
   sudo cp /home/seed/Desktop/OpenSSLLab/Makefile/home/seed
   /openssl-1.0.1/working
   ```

2.
   ```
   sudo cp /home/seed/Desktop/OpenSSLLab/serv.cpp/home/seed
   /openssl-1.0.1/working
   ```

3.
   ```
   sudo -i
   cd /home/seed/openssl-1.0.1/working/
   nc.openbsd -l 4444 > client.csr
   ```
   Start a netcat session in order to move the client CSR.

On the Client VM:

1. `nc.openbsd -w 3 192.168.1.101 4444 < client.csr`
2. `cp /home/seed/Desktop/OpenSSLLab/Makefile/home/seed /openssl-1.0.1/working`

On the Server VM:

1. ca sign csr: `openssl ca -in client.csr -out client.crt -cert ca.crt -keyfile ca.key -config /usr/lib/ssl/openssl.cnf`
   Enter the credentials necessary. Now our client has a signed certificate!

On the Client VM:

1. `sudo -i`
2. `cd /home/seed/openssl-1.0.1/working/`
3. `nc.openbsd -l 4444 > client.crt`

On the Server VM:

1. `nc.openbsd -w 3 192.168.1.100 4444 < client.crt`
   Now the client has the signed certificate!

11 Challenge 1: Setup Server <>

For this challenge...

1. Create the server keys;
2. Create the server certificate signing request;
3. Create the server certificate.

Estimated Time: 25 Minutes
Deliverable: A signed server certificate

12 Task 4: Verify Client and Server Setup <>

1. Move the necessary files on the client and server;
2. Modify `cli.cpp` and `serv.cpp`;
3. Modify the Makefile;
4. Verify the trust between server and client.

Estimated Time: 25 Minutes;
Deliverable: A successful connection between the client and server is made.

On the Server VM:
Navigate to the working directory

1. `cd /home/seed/openssl-1.0.1/working`
2. Move the certificate authority's certificate over to the client.
   `nc.openbsd 192.168.1.100 4444 < ca.key`
3. Modify server port to `4444`.
   `sudo gedit serv.cpp`
4. Run the make file once `serv.cpp` has been changed.
   `sudo make`
5. Run `./serv` (Use the server key as the PEM phrase, if that doesn't work, try the CA's PEM phrase).
   ommand: `sudo ./serv`

On the Client VM:
Navigate to the working directory

1. `cd /home/seed/openssl-1.0.1/working`

2. Retrieve the certificate authority's certificate from the server
   `nc.openbsd -l 4444 > ca.key`

3. Modify `cli.cpp` server's ip to the ip of (y)our server (`192.168.1.101`), and change server port to `4444`.
   `sudo gedit cli.cpp`

4. Run the makefile, once `cli.cpp` has be changed
   `sudo make`

5. Run the command: `./cli` (Use the client key as the PEM phrase, if that doesn't work, try the CA's PEM phrase). Run the command: `sudo ./cli`

13 Challenge 2: Comprehending **`cli.ccp`** and **`serv.cpp`** <>

Review the `cli.cpp` and `server.cpp`

1. What are each of the programs doing?
2. Where is the handshaking taking place?
3. What files are necessary for this code to operate as intended?

Estimated Time: 15 Minutes;
Deliverable: Answer the questions.

14 Questions - 2 <>

Q3: What is a Certificate Signing Request and why is it needed?

Q4: Why is a certificate important when creating a server?

Answer to Q3:
The certificate signing request is a certificate request that is signed by the originating party's private key. The purpose of this request is to be used in the creation of an actual certificate that is signed by the CA.

Answer to Q4:
The certificate is important because it verifies the authenticity of the server. The server would likely not be used by many outside users if it had no certificate or way of proving its legitimacy.

15 Information: How SSL is implemented after certificates are made <>

1. The SSL handshake;
2. What is checked on the certificate;
3. Why are session keys used for encryption.

Estimated Time: 15 Minutes;
Deliverable: Discussion/Lecture.

The SSL Handshake:

1. The client browser sends a request to the server to identify itself; [5]
2. The server sends its SSL certificate to the client along with its public key [5];
3. The client browser checks if the certificate is valid and if it is, sends the server a session key that is encrypted with the server's public key [5];
4. The server decrypts the session key and sends an acknowledgment back to the client using the session key [5];
5. The server and client now have an encrypted SSL channel that they will use to communicate with each other [5].

SSL certificate checking

1. Verify the subject of the certificate [5];
2. Verify the certificate is not expired or revoked [5];
3. Verify the certificate is on the list of trusted certificates [5].

16 Task 5: Create a Public/Private Key Pair in RSA and Encrypt a Message <>

1. Encrypt on the Server VM and decrypt on the Client VM;
2. You should make a new directory to store the files;
3. You need to make the private key from scratch;
4. The public key needs to be created from the private key.

Estimated Time: 30 Minutes;
Deliverable: A successful encrypted message on the client - that was encrypted using the server's public key.

On the Server VM:

1. `sudo -i`
2. `cd ..`
3. `cd /home/seed/openssl-1.0.1/working`
4. Make a new directory.
   `sudo mkdir testing` followed by *cd testing*
5. Generate a new private key.
   `sudo openssl genrsa -aes128 -out servpri.key 2048`
6. Create a public key from the private one.
   `sudo openssl rsa -in servpri.key -outform PEM -pubout -out servpub.key`

On the Client VM:

1. `sudo -i`
2. `cd ..`
3. `cd /home/seed/openssl-1.0.1/working`

4. Make a new directory.
   `sudo mkdir testing` followed by `cd testing`

5. Make a message of any length.
   `sudo gedit important.txt`

6. Get the server's public key.
   `nc.openbsd -l 4444 > servpub.key`

On the Server VM:

1. Send the public key over to the Client.
   `nc.openbsd 192.168.1.100 4444 < servpub.key`

On the Client VM:

1. Encrypt the message using the public key.
   ```
   openssl rsautl -encrypt -in important.txt -out important_enc.txt
   -inkey servpub.key -pubin
   ```

17 Challenge 3: Decrypt the message on the Server <>

1. You will need to send the encrypted file over to the server.
2. The command is similar to the encrypt command. Hint: Think of what you **need** for decrypting a file.
3. Opening up the decrypted file will give you the original message.

Estimated Time: 15 Minutes;
Deliverable: The decrypted message on the server.

18 Task 6: Create the keys for creating a Digital Signature <>

1. This is a similar process to how we encrypted a file.
2. You will use similar commands for the creation of the keys.

Estimated Time: 25 Minutes;
Deliverable: A digitally signed message on the Server VM.

On the Client VM:

1. `cd /home/seed/openssl-1.0.1/working/testing`
2. Generate a new private key.
   ```
   sudo openssl genrsa -aes128 -out digipri.key 2048
   ```
3. Create a public key from the private one.
   ```
   sudo openssl rsa -in digipri.key -outform PEM
   -pubout -out digipub.key
   ```
4. Create a message to be signed.
   ```
   sudo gedit criticalinfrastuctureinfo.txt
   ```
5. Sign the message with sha256.
   ```
   openssl dgst -sha256 -sign digipri.key -out
   criticalinfrastuctureinfo.sha256 criticalinfrastuctureinfo.txt
   ```
6. Send the public key, digitally signed message, and the original message to the Server
   ```
   nc.openbsd 192.168.1.101 4444 < criticalinfrastuctureinfo.sha256
   ```
7. `nc.openbsd 192.168.1.101 4444 < criticalinfrastuctureinfo.txt`
8. `nc.openbsd 192.168.1.101 4444 < digipub.key`

On the Server VM:

1. `cd /home/seed/openssl-1.0.1/working/testing`
2. Get the messages from the client.
   ```
   nc.openbsd -l 4444 > criticalinfrastuctureinfo.sha256
   ```
3. `nc.openbsd -l 4444 > criticalinfrastuctureinfo.txt`
4. `nc.openbsd -l 4444 > digipub.key`
5. Verify the contents of the messages.
   ```
   openssl dgst -sha256 -verify digipub.key -signature
   criticalinfrastuctureinfo.sha256 criticalinfrastuctureinfo.txt
   ```

   Test to make sure the verification worked (Change the original message and reattempt the verification)

19 Questions - 3 <>

Q5: Why are signatures useful?

Q6: How can verification be used to detect an attack?

Q7: How can a digital signature be used to better ensure authenticity?

Answer to Q5:
Digital Signature are useful because they verify the integrity of message. In addition, they verify the authenticity of the sender.

Answer to Q6:
If a digital signature is used, the digest included with the message will not match the message sent. This alerts the receiver that the message has been tampered with.

Answer to Q7:
A digital signature ensures authenticity by using asymmetric encryption. A message encrypted with party A's private key can only be decrypted with party A's public key. Thus if party A encrypts a message and sends it to party B, party B can be assured that the message is from party A, if the message can be decrypted with party A's public key.

20 Conclusion

<>

1. OpenSSL has a wide variety of uses!
2. You can now use OpenSSL to create certificates, digital signatures, and encrypt with RSA;
3. We explained how the SSL handshaking works.

21 Appendix <

1. Solutions to Challenges;
2. Network Diagram;
3. Tutorial Setup;
4. Change-log;
5. References.

Solutions to Challenges

1. **Challenge 1: Setup Server**

   (a) `sudo openssl genrsa -aes128 -out server.key 1024`
   Record all of the server key information into a file.

   (b) `sudo openssl req -new -key server.key -out server.csr -config /usr/lib/ssl/openssl.cnf`
   Enter all of server certificate information.

   (c) `sudo openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config /usr/lib/ ssl/openssl.cnf`
   Enter the credentials necessary. Now our server has a signed certificate!

2. **Challenge 2: Exploration of cli.ccp and serv.cpp**

   (a) They are sending communication between each other.

   (b) The handshaking is taking place at the ”get client/server certificate” sections.

   (c) The files needed are listed at the top of the code. It’s very easy/intuitive to find.

3. **Challenge 3: Decrypt the message on the Server**

   (a) You will need to send the encrypted file over to the server. On the server:

      i. `nc.openbsd -l 4444 > important_enc.txt`

   On the client:

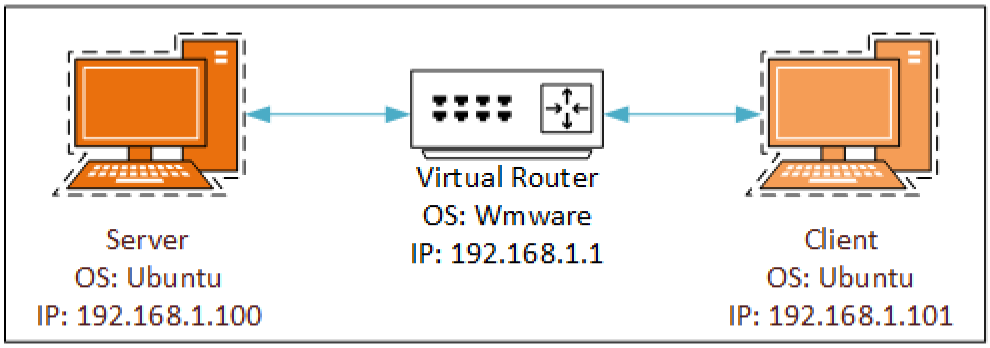
      i. `nc.openbsd 192.168.1.101 4444 < important_enc.txt`

   On the server:

      i. `openssl rsautl -decrypt -in important_enc.txt -out important_dec.txt -inkey servpri.key`

Network Diagram

The networking is straightforward for this tutorial. We need an Ubuntu OS-based server and client. Please note that this tutorial was prepared using Ubuntu 12.04 LTS version of the OS [1]. Both of these machines should be on the same network. Listed are the IP used in the tutorial.



[1]However, we expect no issues to occur with replicating the same tutorial with Ubuntu OS versions greater than 12.04 LTS.

Tutorial Setup

In this tutorial we utilized 2 VMs: a SEED Ubuntu 12.04 client and a SEED Ubuntu 12.04 server. However, 2 vanilla Ubuntu 12.04 machines would be perfectly fine to use. In order to obtain the disk image (ISO) for 12.04 please visit:

`http://old-releases.ubuntu.com/releases/12.04.1/`

Then select the 64 or 32 bit system as appropriate for your OS, and the version 12.04 of Ubuntu. The links on that page should be direct downloads for the OS. Once the OS has been installed and there are 2 VMs created, a few files need to be moved onto both the server and client VM. Of the included files, which can be retrieved from the SEED Labs page, we will need to place the Makefile on both VMs.

In addition, the server VM should have the `serv.cpp` file and the client should have the `cli.cpp` file. These files can be placed on the desktop; inside a folder called "OpenSSL Lab". After this process, please set static IPs and network the VMs as shown in the Network Diagram section of this document. That is it. You are ready to get started!

Change-Log

Change(s)	Contributor(s)	Effective Date
Created the latex document outline	Jonathan Buch	April 22nd, 2017
Started appendix and references	Matthew Kirkland	April 24th, 2017
Added in questions, reviewed challenges	Jonathan Buch	April 24th, 2017
Added news and hardware & software requirements	Matthew Kirkland	April 24th, 2017
Edited the challenges and included step-by-step information for the tasks	Jonathan Buch	April 26th, 2017
Minor fixes, added network diagram, updated and finished news	Matthew Kirkland	April 30th, 2017.
Modified some of the challenges to tasks, restructured existing tasks to incorporate client and server interactions	Jonathan Buch	May 1st, 2017
Minor fixes, updated news, fixed slide links, added question answers	Matthew Kirkland	May 3rd, 2017.
Corrected typos and mistakes in tasks and challenges	Jonathan Buch	May 4th, 2017
Major latex fixes, fixed slide links, added information on SSL Handshaking, added final question answers	Matthew Kirkland	May 5th, 2017.
Added to the tutorial setup section and modified the licensing section	Matthew Kirkland	May 10th, 2017.
Cleaned up spelling and grammar errors for publication	Jonathan Buch	May 11th, 2017
Standardization (edits, consistency, TeX markup cleaning, and more)	Ananth Jillepalli	Aug. 17th, 2017

References

[1] K. Marianne McGee, ”Regulators Warn of Man-in-the-Middle Attack Risks”, last accessed April 30th 2017, http://www.bankinfosecurity.com/regulators-warn-man-in-the-middle-attack-risks-a-9813, 5th APRIL, 2017.

[2] TechTarget, ”PKI (public key infrastructure)”, last accessed 3rd May 2017, http://searchsecurity.techtarget.com/definition/PKI, 1st NOVEMBER 2014.

[3] Lily H. Newman, ”Encrypted Chat Took Over. Let’s Encrypt Calls, Too”, last accessed April 30th 2017, https://www.wired.com/2017/04/encrypted-chat-took-now-encrypted-callings-turn/, 21st APRIL, 2017.

[4] Wenliang Du, ”Crypto Lab – Public-Key Cryptography and PKI”, last accessed April 24th 2017, http://www.cis.syr.edu/ wedu/seed/Labs_12.04/Crypto/Crypto_PublicKey/Crypto_PublicKey.pdf, NOVEMBER 2011.

[5] digicert, ”What Is SSL (Secure Sockets Layer) and What Are SSL Certificates?”, last accessed May 5th 2017, https://www.digicert.com/ssl.htm, JANUARY 2017.