ECE 443/518 Fall 2022 - Graduate Project 1

Secure a Website with HTTPS

Report Due: 12/04 (Sun.), by the end of the day (Chicago time)
Late submissions will NOT be graded

2022/10/30: If you haven't done so, please download the project file prjg1-src.tgz in Section III again as it has been updated to correct a configuration issue with Apache.

I. Objective

In this project, you will learn to secure a website with HTTPS connections and craft an attack to study their vulnerabilities.

As such an attack may open doors for more serious attacks, you should complete the attack within our virtual machine and never try it directly on your own and other people's computers.

II. The Apache Web Server

Please refer to the VM setup to setup the VM first. Start the virtual machine and then login using username 'ubuntu' and password 'ubuntu', preferably using PuTTY.

Our virtual machine comes with the Apache web server installed already with an example website. To verify so, first use 'netstat' to see what network ports are currently in use.

```
ubuntu@ece443:~$ netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.15:22            10.0.2.2:52691          ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

The line starting with 'tcp6 0 0 :::80' indicates that the port 80 is currently in use, in our case by Apache.

Then, use 'wget' to obtain the homepage of the example website located at 'localhost', which confirms that Apache works properly.

```
ubuntu@ece443:~$ wget localhost
--2018-10-07 18:37:54--  http://localhost/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11321 (11K) [text/html]
Saving to: 'index.html'

index.html          100%[===================>]  11.06K  --.-KB/s    in 0s

2018-10-07 18:37:54 (161 MB/s) - 'index.html' saved [11321/11321]
```

Unlike a browser that displays the homepage, wget simply stores the content of the homepage into the file 'index.html'. This is sufficient for us since we don't quite care about its content for our project, though you can always check the content through 'less'.

III. Work with Certificates

2022/10/30: If you haven't done so, please download the project file prjg1-src.tgz as below again as it has been updated to correct a configuration issue with Apache.

A HTTPS server need to identify itself to clients. As discussed in our lectures, we need to ask a certification authority (CA) to sign a certificate for the server. However, since no real CA would sign certificates for our project, we have to create a CA by ourselves and sign server certificates as needed.

First, download the project file, extract it, and change into the directory.

```
ubuntu@ece443:~$ wget http://www.ece.iit.edu/~jwang/ece443-2022f/prjg1-src.tgz
...
ubuntu@ece443:~$ tar -zxf prjg1-src.tgz
...
ubuntu@ece443:~$ cd prjg1-src/
ubuntu@ece443:~/prjg1-src$ ls
apache-ssl-localhost.conf  ca.cnf  create-ca.sh  index.txt
localhost.cnf  localhost-csr.sh  serial.txt  sign-localhost.sh
```

A CA can be created via 'create-ca.sh' that uses 'openssl',

```
ubuntu@ece443:~/prjg1-src$ ./create-ca.sh
Generating a 2048 bit RSA private key
.....................................................+++
.....................................+++
writing new private key to 'ece443-CA.key'
-----
```

Upon completion, two files are generated:

- ece443-CA.key: this is the private key of our CA.
- ece443-CA.pem: this is the certificate of our CA, which contains the public key and other information of our CA signed by our CA.

The identity of our server at 'localhost' can be created via 'localhost-csr.sh' that also uses 'openssl',

```
ubuntu@ece443:~/prjg1-src$ ./localhost-csr.sh
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'ece443-localhost.key'
-----
```

Upon completion, two more files are generated:

- ece443-localhost.key: this is the private key of our server.
- ece443-localhost.csr: this is a certificate signing request (CSR) that we need to send to a CA to sign a certificate for our website. Note that a CSR should contain the public key and the domain name of the server. You may find the domain name of our server at the last line of 'localhost.cnf', which is used by 'localhost-csr.sh'.

The CA then signs the CSR to issue the server certificate 'ece443-localhost.pem'.

```
ubuntu@ece443:~/prjg1-src$ ./sign-localhost.sh
Using configuration from ca.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :ASN.1 12:'IL'
localityName          :ASN.1 12:'Chicago'
organizationName      :ASN.1 12:'IIT'
commonName            :ASN.1 12:'ece443.localhost'
Certificate is to be certified until Oct  8 02:07:30 2019 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
```

IV. Enable HTTPS Connections

With all certificates and keys being created, we are ready to enable HTTPS connections.

First, enable SSL/TLS support in Apache.

```
ubuntu@ece443:~/prjg1-src$ cd /etc/apache2/mods-enabled/
ubuntu@ece443:/etc/apache2/mods-enabled$ sudo ln -s ../mods-available/ssl.* .
[sudo] password for ubuntu:
ubuntu@ece443:/etc/apache2/mods-enabled$ sudo ln -s ../mods-available/socache_shmcb.load .
```

Note that when the 'sudo' command ask you for the password, you should type 'ubuntu' and then 'Enter'.

Then, Apache needs to know our intent to enable HTTPS connections. An Apache configuration file 'apache-ssl-localhost.conf' is provided for your convenience. The private key and the (signed) certificate of our server are both referred to in this file. We will need to copy it to Apache's configuration directory and to restart Apache so it will see the changes.

```
ubuntu@ece443:/etc/apache2/mods-enabled$ cd ~/prjg1-src/
ubuntu@ece443:~/prjg1-src$ sudo cp apache-ssl-localhost.conf /etc/apache2/sites-enabled/
ubuntu@ece443:~/prjg1-src$ sudo service apache2 restart
```

Now, verify that the HTTPS port 443 is in use and then fire 'wget https://localhost'.

```
ubuntu@ece443:~/prjg1-src$ netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.15:22            10.0.2.2:54065          ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::443                  :::*                    LISTEN
ubuntu@ece443:~/prjg1-src$ wget https://localhost
--2018-10-07 21:23:46--  https://localhost/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)::1|:443... connected.
ERROR: cannot verify localhost's certificate, issued by 'CN=ece443,OU=ECE,O=IIT,L=Chicago,ST=IL,C=US':
  Unable to locally verify the issuer's authority.
To connect to localhost insecurely, use '--no-check-certificate'.
```

Clearly, the HTTPS port 443 is in use and Apache presents wget with the certificate of our server. However, wget complains that it cannot verify the certificate. This is as expected since wget has no knowledge of our CA.

Obviously we do not want to connect to localhost insecurely via the option '--no-check-certificate'. Instead, we tell wget to trust our CA by providing the certificate of our CA via the option '--ca-certificate'.

```
ubuntu@ece443:~/prjg1-src$ wget https://localhost --ca-certificate=ece443-CA.pem
--2018-10-07 21:33:48--  https://localhost/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)::1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11321 (11K) [text/html]
Saving to: 'index.html'

index.html                          100%[=================================================================>]

2018-10-07 21:33:48 (69.8 MB/s) - 'index.html' saved [11321/11321]
```

No more complains! wget believes that our server is secure.

V. The Attack

We will now study the vulnerabilities in HTTPS connections by crafting an attack in our virtual machine. The objective is to fool wget to believe a HTTPS website at the domain name 'ece443.hacked' to be secure. We will reuse the example website as well as our CA.

First of all, you will need to generate a certificate for the server at 'ece443.hacked'. Refer to Section III on how to achieve this. As a hint, you will need to modify 'localhost.cnf', 'localhost-csr.sh', and 'sign-localhost.sh'. Make copies before any modification.

Second, you will need to point the domain name 'ece443.hacked' to 'localhost' via DNS spoofing. Since we don't want to setup or modify any DNS server, we will achieve this by modifying the file '/etc/hosts' that all DNS queries will consult first. You will need to add one line '127.0.0.1 ece443.hacked' to the end of the file. Note that since this file is a system file, you will also need to use 'sudo' to access it, e.g. 'sudo vim /etc/hosts'. You may use 'ping' to confirm this is done correctly.

```
ubuntu@ece443:~/prjg1-src$ ping ece443.hacked
PING ece443.hacked (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.028 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.029 ms
^C
--- ece443.hacked ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.020/0.025/0.029/0.007 ms
```

Finally, modify 'apache-ssl-localhost.conf' to refer to the private key and the (signed) certificate of the server at 'ece443.hacked', copy it to Apache's configuration directory again, and restart Apache. Run 'wget https://ece443.hacked --ca-certificate=ece443-CA.pem' to verify that the attack actually works.

VI. Deliverables

Submit the following to Blackboard for this project.

1. Private keys and certificates for the CA, the localhost server, and the ece443.hacked server.
2. Screenshots of failed and successful wget executions on the localhost server, and of successful wget execution on the ece443.hacked server, all with the timestamps.
3. A project report, which should include a summary of your experimental setup and a discussion of your findings. In particular, please address the items below (you may need to perform additional research online).
   - Consider the four files: 'ece443-CA.key', 'ece443-CA.pem', 'ece443-localhost.key', 'ece443-localhost.pem'. Which one is the secret of the CA? Which one is the secret of the server? Which one(s) should be released to public? Why?
   - Run 'wget https://www.google.com' in the VM. Does wget complain? Where is the CA of google's server certificate located in the VM?
   - What is the purpose of the file '/etc/hosts'? Where is '/etc/hosts' located in your own computer? (Yes, Windows and MacOS both use that file too.) Check that file and see if there is anything unusual there.

The project should be done individually. You can discuss the project with other students but all the source code and writings should be your OWN. PLAGIARISM and called for DISCIPLINARY ACTION. NEVER share your source code and reports with others.