Certificate Authority

Section 2 - Team 2

CPRE 431

<u>Team members</u>

Jeremy Galang

Veronica Torres

Priyanka Kadaganchi

Catherine Samatas

Swechha Ghimire

General Overview

Our team agreed to work on the CA and it’s possible threats or attacks that are executed to breach the CA’s security policies. CA is a certificate authority which is a well-trusted, third-party digital certificate issuing body. This digital certificate authenticates the public key's ownership by the certificate's named subject. This mechanism is used such that end-users can only access pages checked by a reputable certificate by the intended certificate authority.

The SSL (Secure Socket Layer) certificate is the most common form of a digital certificate. This certificate links the owner’s data of a web server to a cryptographic key that is subject-generated. In the SSL/TLS protocol, to begin a secure session between the browser and SSL certificate hosting web server, the keys are used . The SSL certificate must contain the website’s literal domain name issued by a trusted certificate authority (not expired) for it to be able to access a web server successfully without any security alerts.

Over the past 15 years, certificate authorities have been popularized and are now the industry standard for remote network connectivity. Also, since the subject has a private key used to produce a digital signature, they are considered safe for authenticating subject identification. While CA’s are not a threat to the network security, if they are compromised, they will act like an extreme danger entering through the trustworthy CA scheme, wrong certificates may be provided that to end-users will look like legitimate certificates. Hacks are regularly done and it is mostly working with the SQL injection which obtains access to several trustable CAs, with most massive certificate authority breach in 2011.

There are some right conditions in which the certificate authorities can generate malicious digital certificates. The assignment here mainly talks on the risks of not using trusted certificate authorities. A standard end-user, intruder, site server, and third-party certificate authority are included in our implemented scheme.

A client that is an ordinary person who is browsing the web. The user wants to access the web server which has a signed certificate. A signed certificate was issued by the website from the CA. The attacker has a web server that manipulates and runs a copy of that website or a fake one, and since it is malicious it doesn’t have a certificate which is from a trusted certificate authority.

Group Contributions

Name	% of contribution	Contribution
Jeremy Galang	20	Performed the experiment. Was able to help set up the LAMP server, enable SSL to Apache, install browser to work through X11, and setup Certificate authority.
Veronica Torres	20	Performed the experiment. Helped setup LAMP server, install browser to work through X11, setup Certificate authority, and generate certificate sign requests.
Priyanka Kadaganchi	20	Worked on the general overview, introduction and conclusion. Contributed to the report by researching what is the certificate authority, how it works and how we worked on this project, what possible implementations we might be performing.
Catherine Samatas	20	Worked on the other security issues of certificate authority as well as added input in the general overview and defence. My contribution was informed by research.
Swechha Ghimire	20	Worked on parts of the security issues of the certificate authority that was related to defend against threats and vulnerabilities related to our topic. Contributed to research of the defense component of our project as well.

Introduction

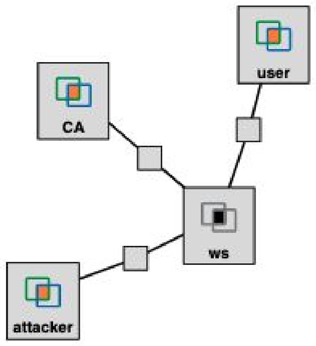
In our project we attempted to showcase the effect of an attacker stealing a certificate authority's private key to forge fake certificates and use it to trick the user into thinking the site's certificate is legitimate. Before doing so, we will need to set up a LAMP server on a web server and set up the certificate authority. We will need to understand how to generate the files and keys necessary for creating certificates.

Approach (About the attack performed + Technique)

An attacker can steal a certificate authority's private keys and forge certificates to act as the certificate authority. They would connect to the CA's system, and then attempt to make a copy or steal the private key and store it on their machine to generate their own certificates.

Before we get into our attack technique, we will first discuss our procedure for setting up our environment.

We created 3 nodes in GENI:



- **CA** - acts as the "certificate authority"
- **Attacker** - individual attacking the system
- **WS** - web server
- **User** - a regular user utilizing the web server

First, we needed to set up the LAMP server on the **web server**.

We needed MySQL, Apache, and PHP (with a few extra packages), so we installed them using:

```
$ sudo apt-get install mysql-server
$ sudo apt-get install apache2
$ sudo apt-get install php libapache2-mod-php
```

We ran into errors below with the additional PHP packages. We found that there were some errors regarding packages being out of date, missing, or deprecated. We were able to resolve this by individually attempting to run each package and finding the most up to date versions.

```
$ sudo apt-get install php-pear php-fpm php-dev php-zip php-curl php-xmlrpc php-gd php-mysql
php-mbstring php-xml libapache2-mod-php
$ sudo apt-get install php-intl php-imagick php-imap php-mcrypt php-memcache php7.0-ps
php-pspell php-recode php-snmp php7.0-sqlite php-tidy php7.0-xsl
```

Next, we connected to the **user** node and installed firefox in the terminal using:

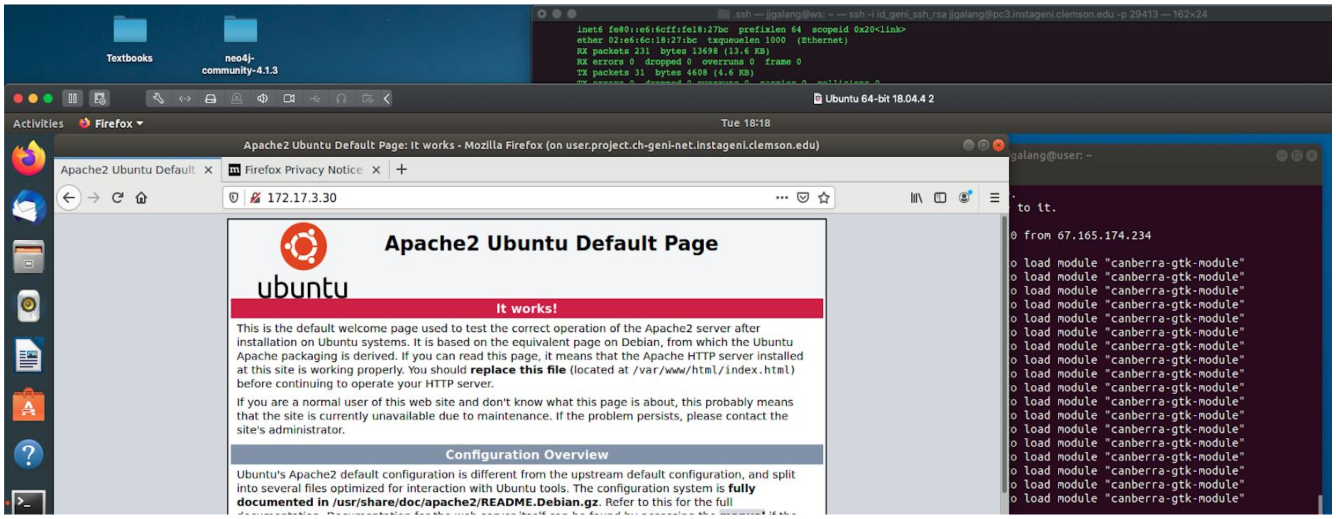
```
$ sudo apt-get install firefox
```

Here, we used a technique known as X11 Forwarding, this essentially allows users to run GUI programs via SSH. However, we could not use the traditional tools such as Powershell/Terminal to connect to the server. For Mac, we had to use XQuartz and on Windows we had to use PuTTy.

When connecting to the **user** node, we needed to add an extra flag (or switch) to our command.

```
$ ssh -YC -i id_geni_rsa <Netid>@<hostname>.edu -p <port number>
```

The "YC" flag indicates it requires a display for the GUI application to show and enable compression. The compression was very important because X11 incurs a lot of overhead. Now we access the Apache website created by the **web server**.



Textbooks
neo4j-community-4.1.3
.ssh — jjgalang@ws: ~ — ssh -i id_geni_ssh_rsa jjgalang@pc3.instageni.clemson.edu -p 29413 — 162×24
Ubuntu 64-bit 18.04.4 2
Activities
Firefox
Tue 18:18
Apache2 Ubuntu Default Page: It works - Mozilla Firefox (on user.project.ch-geni-net.instageni.clemson.edu)
Apache2 Ubuntu Default
Firefox Privacy Notice
172.17.3.30
galang@user: ~

```
inet6 fe80::e6:6cff:fe18:27bc  prefixlen 64  scopeid 0x20<link>
ether 02:e6:6c:18:27:bc  txqueuelen 1000  (Ethernet)
RX packets 231  bytes 13698 (13.6 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 31  bytes 4608 (4.6 KB)
```

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

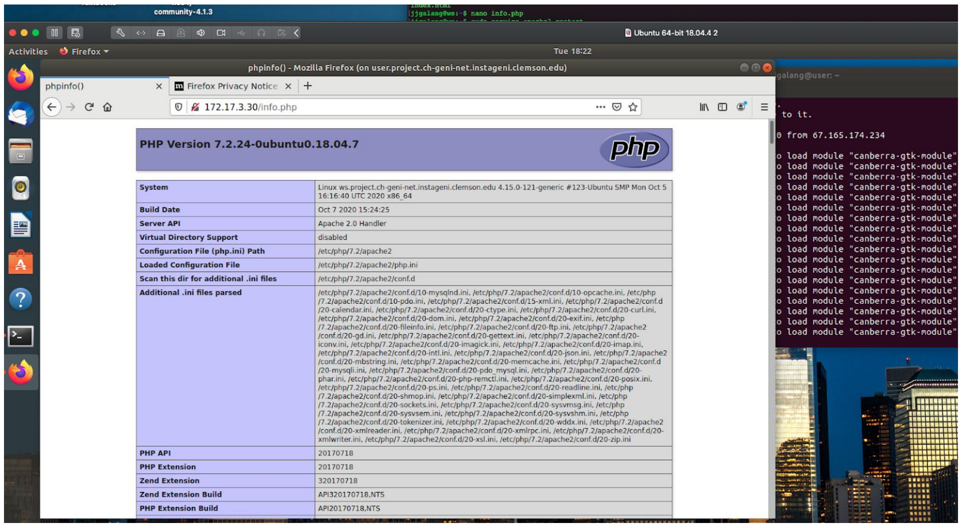
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full

```
 to it.

0 from 67.165.174.234

o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
o load module "canberra-gtk-module"
```

We added a simple PHP script to demonstrate the functionality and packages were installed correctly:



phpinfo() - Mozilla Firefox (on user.project.ch-geni-net.instageni.clemson.edu)

172.17.3.30/info.php

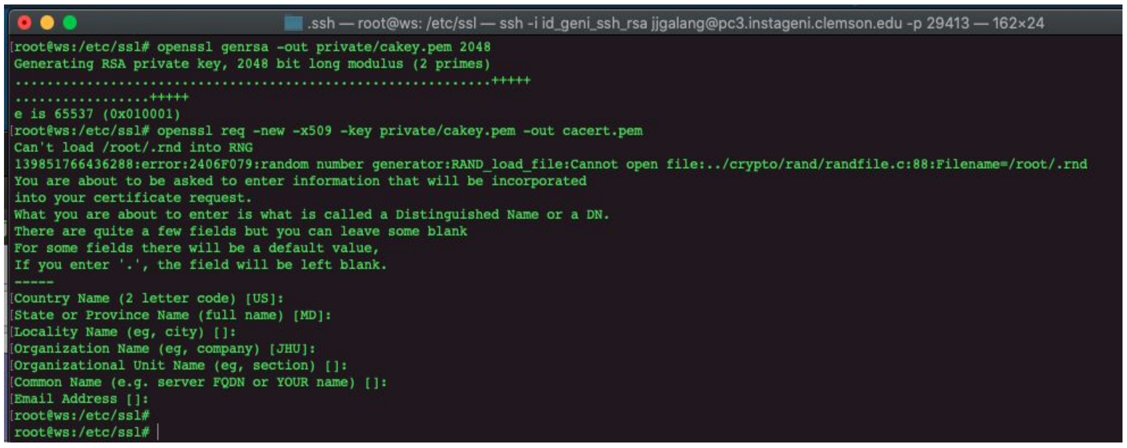
PHP Version 7.2.24-0ubuntu0.18.04.7

System	Linux ws.project.ch-geni-net.instageni.clemson.edu 4.15.0-121-generic #123-Ubuntu SMP Mon Oct 5 16:16:40 UTC 2020 x86_64
Build Date	Oct 7 2020 15:24:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/15-xml.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-dom.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-imagick.ini, /etc/php/7.2/apache2/conf.d/20-imap.ini, /etc/php/7.2/apache2/conf.d/20-intl.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mbstring.ini, /etc/php/7.2/apache2/conf.d/20-memcache.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-php-remctl.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-ps.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-simplexml.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-wddx.ini, /etc/php/7.2/apache2/conf.d/20-xmlreader.ini, /etc/php/7.2/apache2/conf.d/20-xmlrpc.ini, /etc/php/7.2/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.2/apache2/conf.d/20-xsl.ini, /etc/php/7.2/apache2/conf.d/20-zip.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS

Now on the **web server**, we had to modify the openssl.cnf file. We had to modify the directory to be **/etc/ssl/**, change default country name to **US**, province name to **MD**, organization to **JHU**, and Organizational unit name to **ISI**.

Here, we generate the private key for the certificate authority. **This is important, because this is what the attacker will be "stealing" in order to forge their own certificate**!

In the /etc/ssl directory, we created a *newcerts* folder, and *index.txt* file to help generate our private key. We will use the command:
$ openssl genrsa -out private/cakey.pem 2048

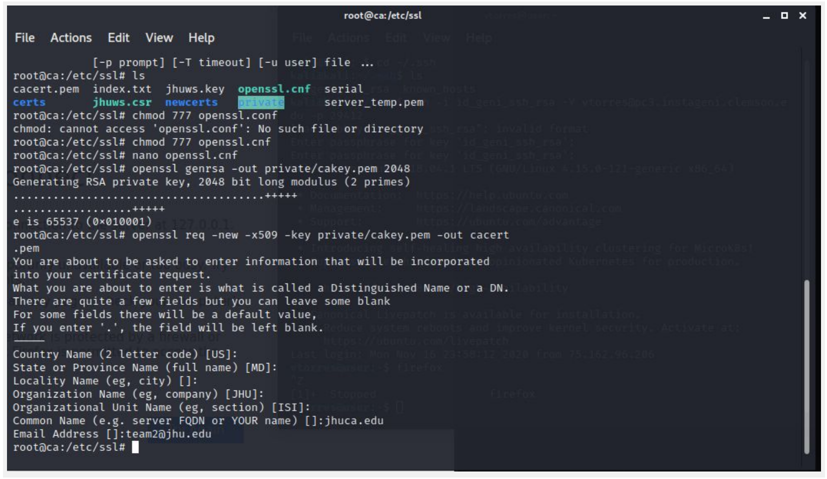


.ssh — root@ws: /etc/ssl — ssh -i id_geni_ssh_rsa jjgalang@pc3.instageni.clemson.edu -p 29413 — 162×24

```
root@ws:/etc/ssl# openssl genrsa -out private/cakey.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..........................................................+++++
................+++++
e is 65537 (0x010001)
root@ws:/etc/ssl# openssl req -new -x509 -key private/cakey.pem -out cacert.pem
Can't load /root/.rnd into RNG
139851766436288:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/rand/randfile.c:88:Filename=/root/.rnd
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [MD]:
Locality Name (eg, city) []:
Organization Name (eg, company) [JHU]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
root@ws:/etc/ssl#
root@ws:/etc/ssl# 
```

Then we created the root digital certificate for the certificate authority by executing the following command:

$ openssl req -new -x509 -key private/cakey.pem -out cacert.pem



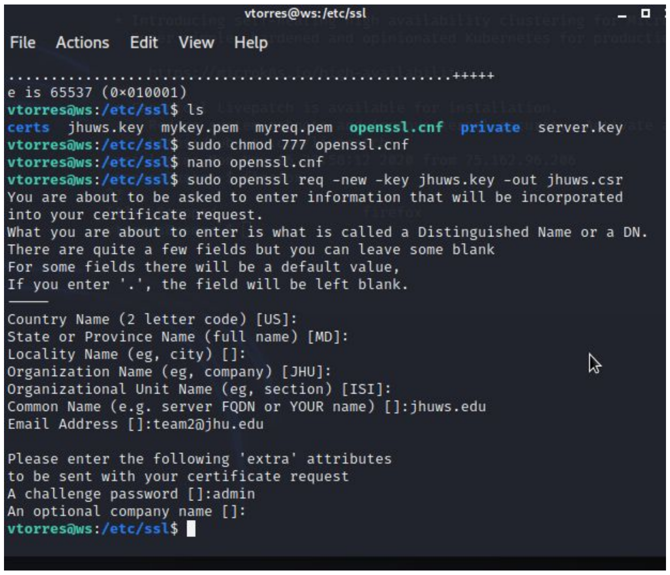
```
File   Actions   Edit   View   Help
        [-p prompt] [-T timeout] [-u user] file ...
root@ca:/etc/ssl# ls
cacert.pem  index.txt  jhuws.key  openssl.cnf  serial
certs       jhuws.csr  newcerts   private      server_temp.pem
root@ca:/etc/ssl# chmod 777 openssl.conf
chmod: cannot access 'openssl.conf': No such file or directory
root@ca:/etc/ssl# chmod 777 openssl.cnf
root@ca:/etc/ssl# nano openssl.cnf
root@ca:/etc/ssl# openssl genrsa -out private/cakey.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.......................................+++++
....................+++++
e is 65537 (0×010001)
root@ca:/etc/ssl# openssl req -new -x509 -key private/cakey.pem -out cacert
.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [MD]:
Locality Name (eg, city) []:
Organization Name (eg, company) [JHU]:
Organizational Unit Name (eg, section) [ISI]:
Common Name (e.g. server FQDN or YOUR name) []:jhuca.edu
Email Address []:team2@jhu.edu
root@ca:/etc/ssl#
```

Next we generated a certificate sign request on the web server.

First we generated a private key for the web server using:
$ sudo openssl genrsa -out jhuws.key 2048

Then we configured the openssl.cnf file like we did previously for the ca node. After that we generated a certificate sign request for the webserver:

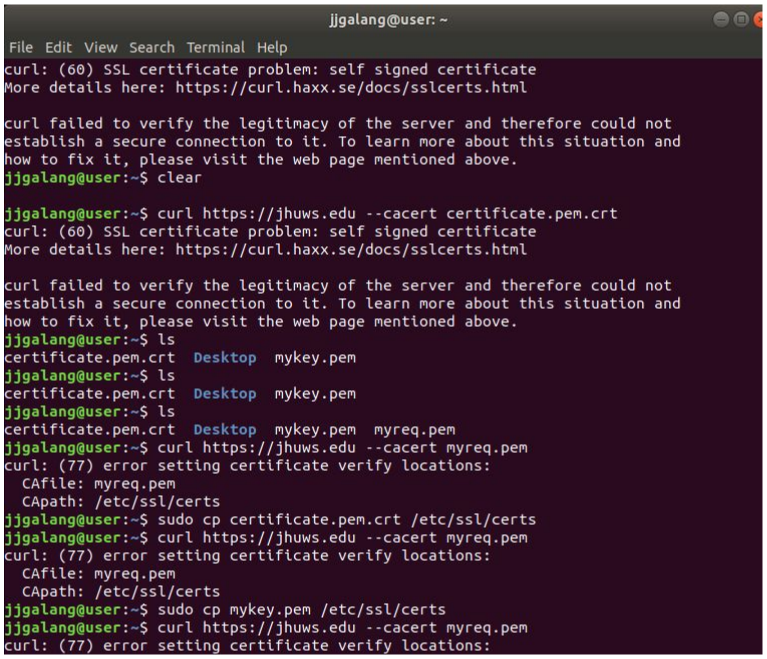
$ sudo openssl -req -new -key jhuws.key -out jhuws.csr



```
File   Actions   Edit   View   Help
..................................................+++++
e is 65537 (0×010001)
vtorres@ws:/etc/ssl$ ls
certs  jhuws.key  mykey.pem  myreq.pem  openssl.cnf  private  server.key
vtorres@ws:/etc/ssl$ sudo chmod 777 openssl.cnf
vtorres@ws:/etc/ssl$ nano openssl.cnf
vtorres@ws:/etc/ssl$ sudo openssl req -new -key jhuws.key -out jhuws.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [MD]:
Locality Name (eg, city) []:
Organization Name (eg, company) [JHU]:
Organizational Unit Name (eg, section) [ISI]:
Common Name (e.g. server FQDN or YOUR name) []:jhuws.edu
Email Address []:team2@jhu.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:admin
An optional company name []:
vtorres@ws:/etc/ssl$
```

Here we ran into issues actually signing the certificate, but we ended up doing another workaround. The certificate didn't appear to sign whatsoever, and we have attempted many workarounds. We tried debugging our environments, reaching out to the TAs for help, verifying the .conf files were modified correctly and everything. We ended up following this guide to essentially generate and self sign our own certificate, but we ran into this problem:



```
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
jjgalang@user:~$ clear

jjgalang@user:~$ curl https://jhuws.edu --cacert certificate.pem.crt
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
jjgalang@user:~$ ls
certificate.pem.crt  Desktop  mykey.pem
jjgalang@user:~$ ls
certificate.pem.crt  Desktop  mykey.pem
jjgalang@user:~$ ls
certificate.pem.crt  Desktop  mykey.pem  myreq.pem
jjgalang@user:~$ curl https://jhuws.edu --cacert myreq.pem
curl: (77) error setting certificate verify locations:
  CAfile: myreq.pem
  CApath: /etc/ssl/certs
jjgalang@user:~$ sudo cp certificate.pem.crt /etc/ssl/certs
jjgalang@user:~$ curl https://jhuws.edu --cacert myreq.pem
curl: (77) error setting certificate verify locations:
  CAfile: myreq.pem
  CApath: /etc/ssl/certs
jjgalang@user:~$ sudo cp mykey.pem /etc/ssl/certs
jjgalang@user:~$ curl https://jhuws.edu --cacert myreq.pem
curl: (77) error setting certificate verify locations:
```

Despite providing the necessary keys and such, we could not access or verify that the certificate was fully working.

How to defend and detect this type of attack

There are other ways to avoid malware from occurring like having an antivirus enabled to stop malicious software installation and execution, the user may discourage malware from operating. Furthermore, if there were a more sophisticated virus than the one, we created that had antivirus resistance techniques, it may still be avoidable by warning the users. Educating users about security threats would encourage them to make smart choices which in turn avoid the hacks, trojan viruses, phishing emails, etc.

The discrepancy between the official website and the fake one would be hard to point at as the digital certificate can check and trust the certificate authorities or possibly it didn’t exist. The reason it’s essential is because the browser will prevent the user from accessing the web in the system where they exist. This exists because a certificate that the browser downloads comes from a legitimate website. This certificate includes encrypted information by the certificate authority's private key and the certificate signs that. The browser would then be decoded by the CA's corresponding public keys. The page is assumed to be valid as long as they line up.

It will be incredibly difficult for an attacker to fake or obtain a targeted web server's private key. For the certificate authority, this would be a severe violation of security that would cause the leaked keys to be illegitimate. When a website detects a discrepancy between the private certificate key of a web server and the public key, or when an unverified source signs the certificate, it considers the domain to be malicious. The browser alerts the absence of a signed certificate and not allow entry to the web site.

As stated, any web service can be breached, and this is when the next measure of safety comes into play. A user can install an add-on that will implement revocation authentication on their browser. Testing for revocation is where the list containing trustworthy certificate authorities' public keys is modified. If a certificate authority's protection was compromised and the private key was maliciously exploited, it will be excluded from the trustworthy list. This acts as the final checkpoint for the web browser before it allows access to a malicious website.

Additional Security Threats

There are various other security issues that can be associated with certificate authority. The first security threat stems from the use of certificate revocation. Certificate revocation often uses a certificate revocation list (CRL) or the Open Certificate Status Protocol (OCSP) to check the revocation status of a certificate. If for some reason the revocation status information is unavailable, the variation defaults to a soft fail open or close. Which means that either the client/user of the certificate is no longer protected or the endpoint cannot be reached. One solution to this security issue is take out certificate revocation and implement short validity periods.

Another major security issue of certificate authority can occur with root store certificates which references certificate authorities that automatically sign other certificate authorities. Root certificates are common (a single computer can have 1000s) and they can be preinstalled with a computer or user installed. Some of these certificates can allow entry points for hackers specifically allowing a man in the middle attack (MITM). An example of this is if a hacker put up a fake website imitating amazon that uses the specific certificate that is presigned. The pre signed certificate would be unaware the request is fake and the user of the fake website would continue unaware of the hack. In recent history there have been major PC companies that distributed computers with root store certificates that give access for the MITM attacks. The first was in 2014 with Lenovo who included a root certificate with the intent of allowing some advertising capabilities. This bug was distributed in computers for over a half a year. The same idea then happened again with Dell in 2015. The defence for this security issue is manually going through the root store certificates One way to defend against these root certificates is to manually verify they are what is expected. However this solution is not feasible in most cases due to the sheer number of certificates that can exist. The process of filtering through the certificates can speed up the use of tools.

Conclusion

Lastly for the conclusion our team understood the importance of the certificate authority and how simple an attack can be done if not present. However, the more vital lesson was how essential the creation of the CA was. This is sometimes ignored being a regular user and is an afterthought. Half of the population doesn't care about all this so they have no idea about these risks. When the ordinary user visits a domain, they should be able to trust it as per given by the SAC (Service Certificate Authority). A CA's system and website need to be fully secured in order to prevent attackers from stealing a private key. It is a daunting job to undertake to undermine a CA so that it can be deemed trustworthy. It would be difficult to tell if a website were the website you thought it would be without certification authority.

Lessons learned

From a system defender's perspective, we found that it was important to follow the default deny security policy mentioned in class. Any unrecognized/unverified services should be denied by default. For example, no unknown user should have SSH access to the CA's server and gain unauthorized access to the private key that allows the generation of certificates. From an attacker's perspective, these forged certificates could be used for phishing or social engineering users. This would trick users into thinking that the website is legitimate and their information is safe. From an overall perspective, we were able to gain an understanding of how generating and signing certificates worked using OpenSSL. We were also able to learn how to setup and install packages to create a LAMP server.

Source Cited

The lab itself: (For guiding us and showing us how to setup our environment)
http://mountrouidoux.people.cofc.edu/CyberPaths/digital_certificate.html

https://en.wikipedia.org/wiki/Certificate_authority

https://blogg.bekk.no/how-to-sign-a-certificate-request-with-openssl-e046c933d3ae

https://www.ssls.com/blog/why-ssl-certificate-revocation-checks-dont-always-work-like-they-should/

J. A. Berkowsky and T. Hayajneh, "Security issues with certificate authorities," *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, New York, NY, 2017, pp. 449-455, doi: 10.1109/UEMCON.2017.8249081.