HW 7: Security

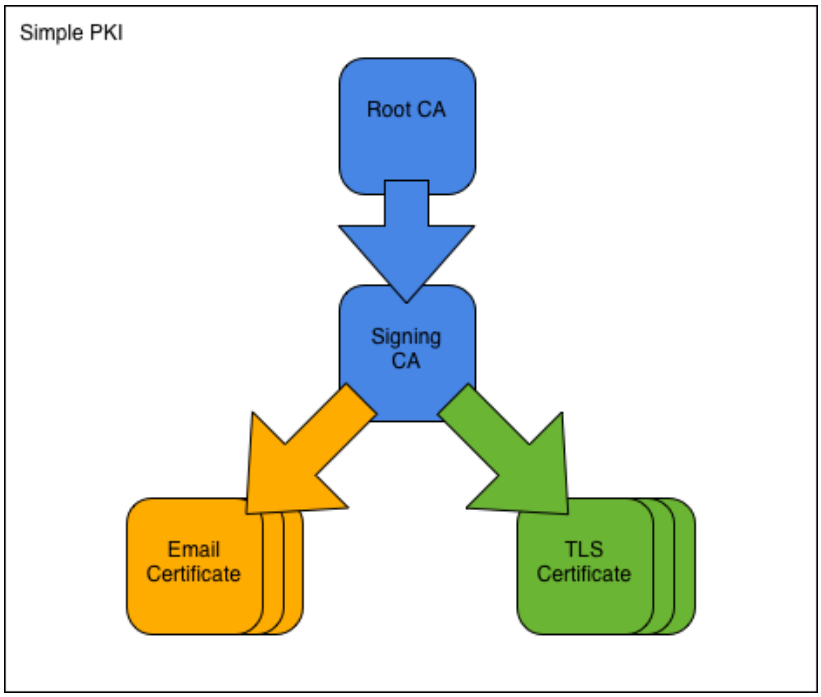
Awesome Pawesome

Team: Tyler Bruno, Shivangi Agarwal, Inderjit Bassi, Lance Barrett, Ryan Moore

Github Link: https://github.com/Hapa1/HW7Security

1: Building a PKI Infrastructure

We want to build a PKI Infrastructure with a Root CA, Signing CA, and TLS certificate as such:



1.1 Clone simple PKI example files

```
hapa1@hapa1-VirtualBox:~$ cd Desktop
hapa1@hapa1-VirtualBox:~/Desktop$ git clone https://bitbucket.org/stefanholek/pk
i-example-1
Cloning into 'pki-example-1'...
remote: Counting objects: 48, done.
remote: Compressing objects: 100% (37/37), done.
remote: Total 48 (delta 20), reused 0 (delta 0)
Unpacking objects: 100% (48/48), done.
Checking connectivity... done.
hapa1@hapa1-VirtualBox:~/Desktop$ cd pki-example-1
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.2 Create directories and database

```
hapa1@hapa1-VirtualBox:~/Desktop$ cd pki-example-1
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ mkdir -p ca/root-ca/private ca/root-ca/db crl certs
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ chmod 700 ca/root-ca/private
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ cp /dev/null ca/root-ca/db/root-ca.db
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ cp /dev/null ca/root-ca/db/root-ca.db.attr
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ echo 01 > ca/root-ca/db/root-ca.crt.srl
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ echo 01 > ca/root-ca/db/root-ca.crl.srl
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.3 Create CA request

```
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ openssl req -new \
>     -config etc/root-ca.conf \
>     -out ca/root-ca.csr \
>     -keyout ca/root-ca/private/root-ca.key
Generating a 2048 bit RSA private key
.....................................+++
.....................................................................................................
.................+++
writing new private key to 'ca/root-ca/private/root-ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.4 Create CA Certificate

```
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ openssl ca \
>     -config etc/root-ca.conf \
>     -in ca/signing-ca.csr \
>     -out ca/signing-ca.crt \
>     -extensions signing_ca_ext
Using configuration from etc/root-ca.conf
Enter pass phrase for ./ca/root-ca/private/root-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: May  3 22:18:23 2018 GMT
            Not After : May  2 22:18:23 2028 GMT
        Subject:
            domainComponent           = org
            domainComponent           = simple
            organizationName          = Simple Inc
            organizationalUnitName    = Simple Signing CA
            commonName                = Simple Signing CA
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
                E9:AC:52:31:E9:0B:7F:A2:C9:7C:D0:05:D4:9E:7F:82:29:9C:5E:8A
            X509v3 Authority Key Identifier:
                keyid:23:AB:7D:D8:11:FC:A6:2E:46:BD:AE:7C:C9:43:8E:D0:AC:A0:4F:C6

Certificate is to be certified until May  2 22:18:23 2028 GMT (3652 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.5 Create TLS server request

```
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ SAN=DNS:www.simple.org \
> openssl req -new \
>     -config etc/server.conf \
>     -out certs/simple.org.csr \
>     -keyout certs/simple.org.key
Generating a 2048 bit RSA private key
..................................................+++
.............+++
writing new private key to 'certs/simple.org.key'
-----
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.
-----
1. Domain Component         (eg, com)       []:org
2. Domain Component         (eg, company)   []:simple
3. Domain Component         (eg, pki)       []:
4. Organization Name        (eg, company)   []:Simple Inc
5. Organizational Unit Name (eg, section)   []:
6. Common Name              (eg, FQDN)      []:www.simple.org
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.6 Create TLS certificate

```
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ openssl ca \
>     -config etc/signing-ca.conf \
>     -in certs/simple.org.csr \
>     -out certs/simple.org.crt \
>     -extensions server_ext
Using configuration from etc/signing-ca.conf
Enter pass phrase for ./ca/signing-ca/private/signing-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: May  7 07:53:55 2018 GMT
            Not After : May  6 07:53:55 2020 GMT
        Subject:
            domainComponent           = org
            domainComponent           = simple
            organizationName          = Simple Inc
            commonName                = www.simple.org
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                3B:5C:18:FE:CF:DE:3B:41:C1:A4:07:C4:66:91:D5:76:27:42:EC:34
            X509v3 Authority Key Identifier:
                keyid:E9:AC:52:31:E9:0B:7F:A2:C9:7C:D0:05:D4:9E:7F:82:29:9C:5E:8A

            X509v3 Subject Alternative Name:
                DNS:www.simple.org
Certificate is to be certified until May  6 07:53:55 2020 GMT (730 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

1.7 Revoke certificate and generate certificate revocation list

```
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ openssl ca \
>     -config etc/signing-ca.conf \
>     -revoke ca/signing-ca/01.pem \
>     -crl_reason superseded
Using configuration from etc/signing-ca.conf
Enter pass phrase for ./ca/signing-ca/private/signing-ca.key:
Revoking Certificate 01.
Data Base Updated
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$ openssl ca -gencrl \
>     -config etc/signing-ca.conf \
>     -out crl/signing-ca.crl
Using configuration from etc/signing-ca.conf
Enter pass phrase for ./ca/signing-ca/private/signing-ca.key:
hapa1@hapa1-VirtualBox:~/Desktop/pki-example-1$
```

Part 2: Using the PKI with Tomcat

2.1 Create keystore from PKI

```
hapa1@hapa1-VirtualBox:~/code$ openssl pkcs12 -export -in /home/hapa1/Desktop/pki-example-1/certs/sim
ple.org.crt -inkey /home/hapa1/Desktop/pki-example-1/certs/simple.org.key -out mycert.p12 -name tomca
t -CAfile /home/hapa1/Desktop/pki-example-1/ca/root-ca.crt -caname root -chain
Error unable to get local issuer certificate getting chain.
```

2.2 Edit Tomcat's server.xml to use keystore

```
a1-VirtualBox: /opt/tomcat/conf
GNU nano 2.5.3                                File: server.xml

      This connector uses the NIO implementation. The default
      SSLImplementation will depend on the presence of the APR/native
      library and the useOpenSSL attribute of the
      AprLifecycleListener.
      Either JSSE or OpenSSL style configuration may be used regardless of
      the SSLImplementation selected. JSSE style configuration is used below.
  -->

  <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
             maxThreads="150" SSLEnabled="true" keystoreFile="/home/hapa1/code/keystore.txt" keysto
<!--      <SSLHostConfig>
          <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                       type="RSA" />
      </SSLHostConfig>-->
  </Connector>

  <!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
       This connector uses the APR/native implementation which always uses
       OpenSSL for TLS.
       Either JSSE or OpenSSL style configuration may be used. OpenSSL style
       configuration is used below.
  -->
  <!--
  <Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
             maxThreads="150" SSLEnabled="true" >
      <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
      <SSLHostConfig>
          <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
                       certificateFile="conf/localhost-rsa-cert.pem"
                       certificateChainFile="conf/localhost-rsa-chain.pem"
                       type="RSA" />
      </SSLHostConfig>
  </Connector>
  -->

  <!-- Define an AJP 1.3 Connector on port 8009 -->
  <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />


  <!-- An Engine represents the entry point (within Catalina) that processes
       every request.  The Engine implementation for Tomcat stand alone
       analyzes the HTTP headers included with the request, and passes them
       on to the appropriate Host (virtual host).
       Documentation at /docs/config/engine.html -->

  <!-- You should set jvmRoute to support load-balancing via AJP le :
  <Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
  -->
  <Engine name="Catalina" defaultHost="localhost">
```

2.3 The server is now secure!