CS 4371.251
Group 18
04/26/2022

Levi Jacoby
James Helgren
Brandon Swanwick

Executive Summary

Introduction

A public key infrastructure (PKI) can be defined as “a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption” (Wikipedia).

Problem Statement

Security and safety are incredibly important among any business or organization. Therefore, there must be a way to identify and encrypt any entity in communication with said group.

Proposed Solution

Our proposed solution is to introduce a two-tier PKCS architecture to improve website security, prevent information leaks, and protect against account compromise. To achieve this we have constructed a pks system that will utilize a two-tier certificate authority system to meet these qualifications desired by ACME. This in-house two-tier PKCS system will check against employee IDs, positions, the first and last name of the user accessing certain site features. This is to ensure that information is not leaked, accounts remain protected, and bad actors are prevented from attempting to access and compromise ACME’s website.

The client creates a key pair, which consists of a public and private key. A certificate signing request is sent by the client and received by a (trusted) certificate authority. Part of this request consists of the client’s public key. After receiving the CSR, there are two possibilities that depend on the decision the CA must make. If the CA validates the CSR, a private key is issued and a certificate is signed by the certificate authority to the client to use. Otherwise, the request is rejected. The client will lastly use the signed certificate for the security protocol.

Key components

Root CA - Responsible for providing trusted certificates with public/private key pairs, and the setting of validity periods as well as the ability to revoke certificates. (Self-signed)

Issuing CA - Works as a subordinate of the root certificate because the root certificate has its very own security layers assuming that its keys remain unobtainable. This type of certificate plays a “chain of trust” acting as the bridge between an end entity certificate and a root certificate.

TLS Server Authentication Certificate - In this case the TLS server requires client authentication, thus what is needed for the TLS server certificate is; the personal certificate issued to the server and client both from the Issuing CA, as well as the server’s private key.

TLS Client Authentication Certificate - The personal certificate issued to the client by the Issuing CA, and the clients private key.

Certificate Revocation List (CRL) - A list of digital certificates that have been revoked by the issuing CA before their actual or assigned expiration date. This type of list includes certificates that should no longer be trusted and is used by many endpoints, for example web browsers, to verify if a certificate is valid and trustworthy.

Registration Authority - Responsible for accepting requests for digital certificates and authenticating the person or organization making the request.

It should be noted that both processes of creating client and server authentication certificates follow the initial steps of the handshake. First there are greetings between the client and server that contain multiple pieces of information, including a SSL certificate provided by the server to verify its identity, a list of cipher suites (sets of encryption algorithms) the client supports, a client and server random, and the server’s choice of cipher suite. Next the client validates the server’s identity with the SSL certificate provided by the server. A premaster secret is a random string of bytes provided by the client and is encrypted with the public key. It can only be decrypted with the private key by the server. Session keys are generated by both the client and server using the premaster secret, the client random, and the server random. The handshake is completed after an exchange of encrypted “finished” messages between the server and client.

As-Built Specifications and Project Plan

Root CA

The Root CA is the root signer of Certificates of Authorities. It is kept onsite in an offline server, it signs the Issuing CA's certificate of authority which allows it to issue CA's in its place. If the CA signed by the Root CA expires you will have to have the Root CA sign a certificate of authority for the Issuing CA so that it may continue signing certificates of authorities. This ensures that the Issuing CA always has certificates that are recognized by the Root CA and server administrators.

Issuing CA

The Issuing CA is a CA that issues certificates to end entities. In our case there is no limit on the number of intermediate CA's between the route and end entity certificate although there are certainly best practices. We have a chain of two intermediate certificates, and they are the issuing CA followed below it by either the TLS server authentication certificate or the TLS client authentication certificate. Since the root CA is kept offline and secure and the public portion of the Root CA certificate is included in the subordinate CA (in this case the issuing CA) any end-entity certificate signed by the sub CA. This implies that validation can continue happening even when the Root CA and/or intermediate CA is offline.

Certificate Revocation List

In the case of revoking a particular certificate from a client the use of a CRL comes into play. The web server will contact the CA with the corresponding information of the certificate in question and request to revoke the certificate. Then the CA will update the CRL stored at a CRL location with the information of the certificate in question. Following this the next time the client attempts to connect to a web server and downloads the updated CRL and will see the certificate has been revoked. The browser (client) will then see that the certificate has been revoked and they should not trust the web server in question for whatever reason. Generally this process applies to certificates that have not expired yet.

As-Built Specification

Overview

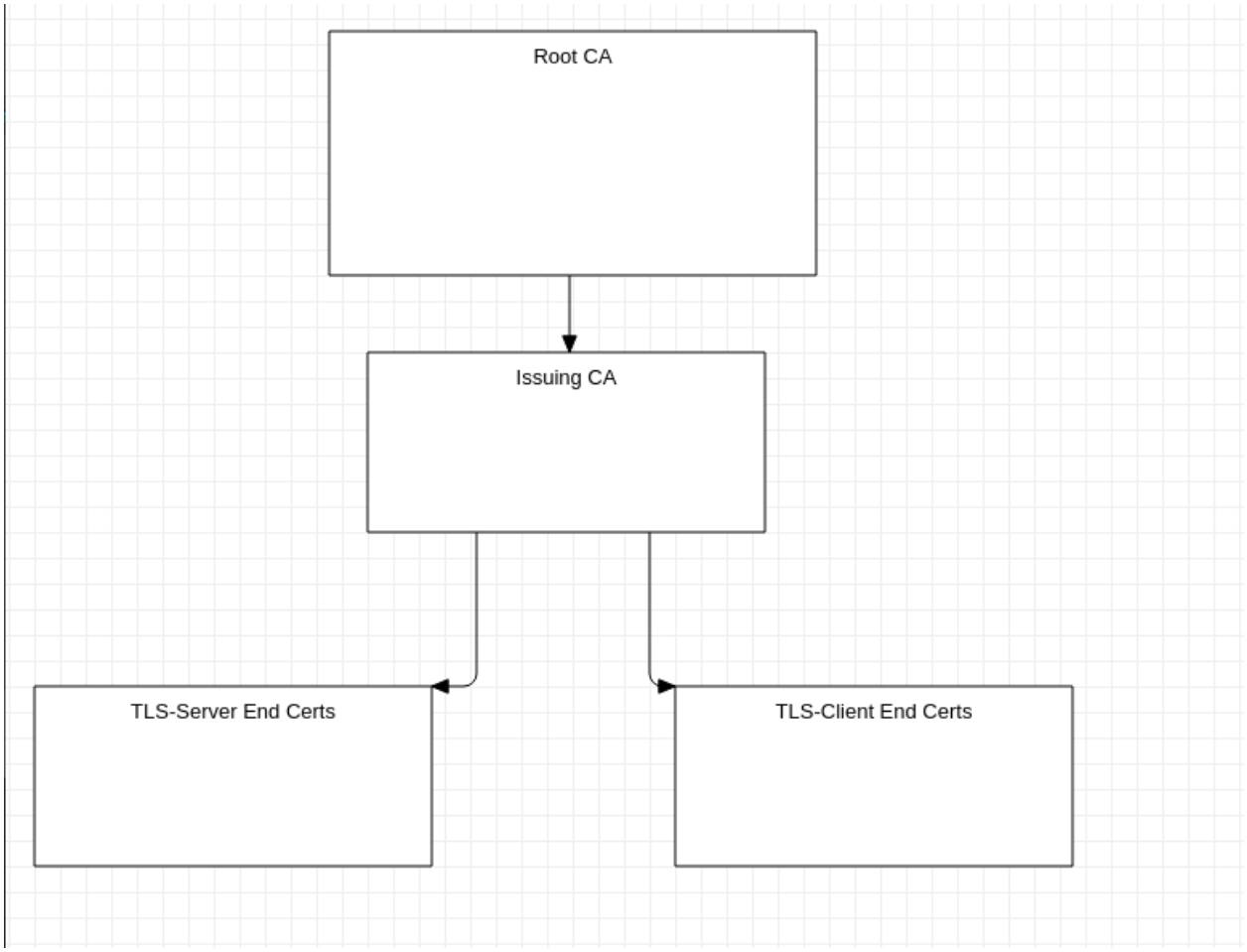
Accomplishments

By implementing a two-tier PKI, we are able to demonstrate how to secure ACME corporation's website and services to protect against malicious actors inside and outside the company.

Next Steps

Considering future threats, we might have to demonstrate how to implement a different PKI infrastructure for further security. We might also need to demonstrate how to recreate this infrastructure once the CA's have passed their expiration date.

General Structure of Two Tier PKI For ACME Corporation



Environment Technical Summary

This system is to be implemented on an ubuntu server. The server will need openssl installed on it. The user implementing this structure must have root privileges so make sure the user implementing this has sudo access.

Installation Instructions

```
leviadmin@leviadmin-VirtualBox:~$ tree ca/
ca/
├── ca
│   ├── issuing-ca
│   │   ├── 01.pem
│   │   ├── 02.pem
│   │   ├── db
│   │   │   ├── issuing-ca.crl.srl
│   │   │   ├── issuing-ca.crl.srl.old
│   │   │   ├── issuing-ca.crt.srl
│   │   │   ├── issuing-ca.crt.srl.old
│   │   │   ├── issuing-ca.db
│   │   │   ├── issuing-ca.db.attr
│   │   │   ├── issuing-ca.db.attr.old
│   │   │   └── issuing-ca.db.old
│   │   └── private
│   │       └── issuing-ca.key
│   ├── issuing-ca.crt
│   ├── issuing-ca.csr
│   ├── root-ca
│   │   ├── 01.pem
│   │   ├── 02.pem
│   │   ├── db
│   │   │   ├── root-ca.crl.srl
│   │   │   ├── root-ca.crl.srl.old
│   │   │   ├── root-ca.crt.srl
│   │   │   ├── root-ca.crt.srl.old
│   │   │   ├── root-ca.db
│   │   │   ├── root-ca.db.attr
│   │   │   ├── root-ca.db.attr.old
│   │   │   └── root-ca.db.old
│   │   └── private
│   │       └── root-ca.key
│   ├── root-ca.crt
│   ├── root-ca.csr
│   └── tls-ca
│       └── private
├── issuing-ca.cnf
├── root-ca.cnf
├── tls-client-cert.cnf
└── tls-server-cert.cnf
```

Directory Tree Structure for the CAs

```
9 directories, 30 files
leviadmin@leviadmin-VirtualBox:~$ tree certs/
certs/
├── ca4371.local.crt
├── ca4371.local.key
├── caD4371.local.key
├── cs4371client.csr
├── cs4371.client.key
├── cs4371.crt
├── cs4371.local.csr
└── TLS-clint.p12
```

Directory Tree Structure for End Certificates.

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl req -new \
> -config root-ca.cnf \
> -out root-ca.csr \
> -keyout ca/root-ca/private/root-ca.key
Generating a RSA private key
.....................................................................................................+++++
.........+++++
writing new private key to 'ca/root-ca/private/root-ca.key'
Enter PEM pass phrase:
```

Creating the root-ca.key file

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca -selfsign -config root-ca.cnf -in ca/root-ca.csr -out ca/root-ca.crt -extensions root_ca_ext
Using configuration from root-ca.cnf
Enter pass phrase for ./ca/root-ca/private/root-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 3 (0x3)
        Validity
            Not Before: Apr 22 21:28:18 2022 GMT
            Not After : Apr 21 21:28:18 2032 GMT
        Subject:
            countryName               = US
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371 Certificate Authority
            commonName                = CS4371 Root CA
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                C4:66:36:3B:8E:EA:BF:33:87:40:D9:90:09:FF:0E:6D:2B:69:E2:44
            X509v3 Authority Key Identifier:
                keyid:C4:66:36:3B:8E:EA:BF:33:87:40:D9:90:09:FF:0E:6D:2B:69:E2:44

Certificate is to be certified until Apr 21 21:28:18 2032 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
```

Signing the root-ca and creating the root-ca.crt file.

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca -gencrl \
> -config root-ca.cnf \
> -out ~/crl/root-ca.crl
Using configuration from root-ca.cnf
Enter pass phrase for ./ca/root-ca/private/root-ca.key:
```

Creating the CRL for the root-ca

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl req -new \
> -config issuing-ca.cnf \
> -out ca/issuing-ca.csr \
> -keyout ca/issuing-ca/private/issuing-ca.key
Generating a RSA private key
.........+++++
............+++++
writing new private key to 'ca/issuing-ca/private/issuing-ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

Creating the issuing-ca

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca \
> -config root-ca.cnf \
> -in ca/issuing-ca.csr \
> -out ca/issuing-ca.crt \
> -extensions signing_ca_ext
Using configuration from root-ca.cnf
Enter pass phrase for ./ca/root-ca/private/root-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4 (0x4)
        Validity
            Not Before: Apr 22 21:33:44 2022 GMT
            Not After : Apr 21 21:33:44 2032 GMT
        Subject:
            countryName               = US
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371 Certificate Authority
            commonName                = CS4371 TLS CA
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
                86:86:2A:FB:92:E4:FC:74:A5:31:BD:C9:B5:15:FC:5F:BC:04:93:EB
            X509v3 Authority Key Identifier:
                keyid:C4:66:36:3B:8E:EA:BF:33:87:40:D9:90:09:FF:0E:6D:2B:69:E2:44

            Authority Information Access:
                CA Issuers - URI:http://pki.cs4371.local/ca/root-ca.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.cs4371.local/ca/root-ca.crl

Certificate is to be certified until Apr 21 21:33:44 2032 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
```

Signing the issuing-ca

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca -gencrl \
> -config issuing-ca.cnf \
> -out ~/crl/issuing-ca.crl
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
leviadmin@leviadmin-VirtualBox:~/ca$
```

Creating the issuing-ca .crl

```
leviadmin@leviadmin-VirtualBox:~/ca$ SAN=DNS:cs4371.local,DNS:www.cs4371.local openssl req -new -config tls-server-cert.cnf -out ~/certs/cs4371.local.csr -keyout ~
/certs/ca4371.local.key
Generating a RSA private key
........+++++
.....................................+++++
writing new private key to '/home/leviadmin/certs/ca4371.local.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. Country Name (2 letters) (eg, US)       []:US
2. State or Province Name   (eg, region)   []:TX
3. Locality Name            (eg, city)     []:Austin
4. Organization Name        (eg, company)  []:CS4371 Section 251
5. Organizational Unit Name (eg, section)  []:CS4371
6. Common Name              (eg, FQDN)     []:Levi
leviadmin@leviadmin-VirtualBox:~/ca$
```

Creating the .csr and .key file for the tls certificate

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca \
> -config issuing-ca.cnf \
> -in ~/certs/cs4371.local.csr \
> -out ~/certs/ca4371.local.crt \
> -extensions server_ext
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 3 (0x3)
        Validity
            Not Before: Apr 22 21:41:12 2022 GMT
            Not After : Apr 21 21:41:12 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = TX
            localityName              = Austin
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371
            commonName                = Levi
        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:
                2D:9C:A5:31:18:D0:AB:B3:82:3A:5C:60:E2:9A:CF:DA:99:24:D1:51
            X509v3 Authority Key Identifier:
                keyid:86:86:2A:FB:92:E4:FC:74:A5:31:BD:C9:B5:15:FC:5F:BC:04:93:EB

            Authority Information Access:
                CA Issuers - URI:http://pki.cs4371.local/ca/issuing-ca.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.cs4371.local/ca/issuing-ca.crl
```

Creating the tls-server certificate

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca \
> -config issuing-ca.cnf \
> -in ~/certs/cs4371.local.csr \
> -out ~/certs/ca4371.local.crt \
> -extensions server_ext
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 3 (0x3)
        Validity
            Not Before: Apr 22 21:41:12 2022 GMT
            Not After : Apr 21 21:41:12 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = TX
            localityName              = Austin
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371
            commonName                = Levi
        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:
                2D:9C:A5:31:18:D0:AB:B3:82:3A:5C:60:E2:9A:CF:DA:99:24:D1:51
            X509v3 Authority Key Identifier:
                keyid:86:86:2A:FB:92:E4:FC:74:A5:31:BD:C9:B5:15:FC:5F:BC:04:93:EB

            Authority Information Access:
                CA Issuers - URI:http://pki.cs4371.local/ca/issuing-ca.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.cs4371.local/ca/issuing-ca.crl
```

Signing the tls-server with the issuing CA

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca -config issuing-ca.cnf -in ~/certs/cs4371-client.csr \
> -out ~/certs/cs4371.crt \
> -policy extern_pol \
> -extensions client_ext
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4 (0x4)
        Validity
            Not Before: Apr 22 21:50:40 2022 GMT
            Not After : Apr 21 21:50:40 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = TX
            localityName              = Austin
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371
            commonName                = Levi
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                58:E8:0B:BA:2C:CE:0D:EE:0A:07:4C:B7:F8:F7:3B:73:4E:12:BE:1C
            X509v3 Authority Key Identifier:
                keyid:86:86:2A:FB:92:E4:FC:74:A5:31:BD:C9:B5:15:FC:5F:BC:04:93:EB

            Authority Information Access:
                CA Issuers - URI:http://pki.cs4371.local/ca/issuing-ca.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.cs4371.local/ca/issuing-ca.crl
```

Creating the tls-client certificates

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca -config issuing-ca.cnf -in ~/certs/cs4371-client.csr \
> -out ~/certs/cs4371.crt \
> -policy extern_pol \
> -extensions client_ext
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4 (0x4)
        Validity
            Not Before: Apr 22 21:50:40 2022 GMT
            Not After : Apr 21 21:50:40 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = TX
            localityName              = Austin
            organizationName          = CS4371 Section 251
            organizationalUnitName    = CS4371
            commonName                = Levi
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                58:E8:0B:BA:2C:CE:0D:EE:0A:07:4C:B7:F8:F7:3B:73:4E:12:BE:1C
            X509v3 Authority Key Identifier:
                keyid:86:86:2A:FB:92:E4:FC:74:A5:31:BD:C9:B5:15:FC:5F:BC:04:93:EB

            Authority Information Access:
                CA Issuers - URI:http://pki.cs4371.local/ca/issuing-ca.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.cs4371.local/ca/issuing-ca.crl
```

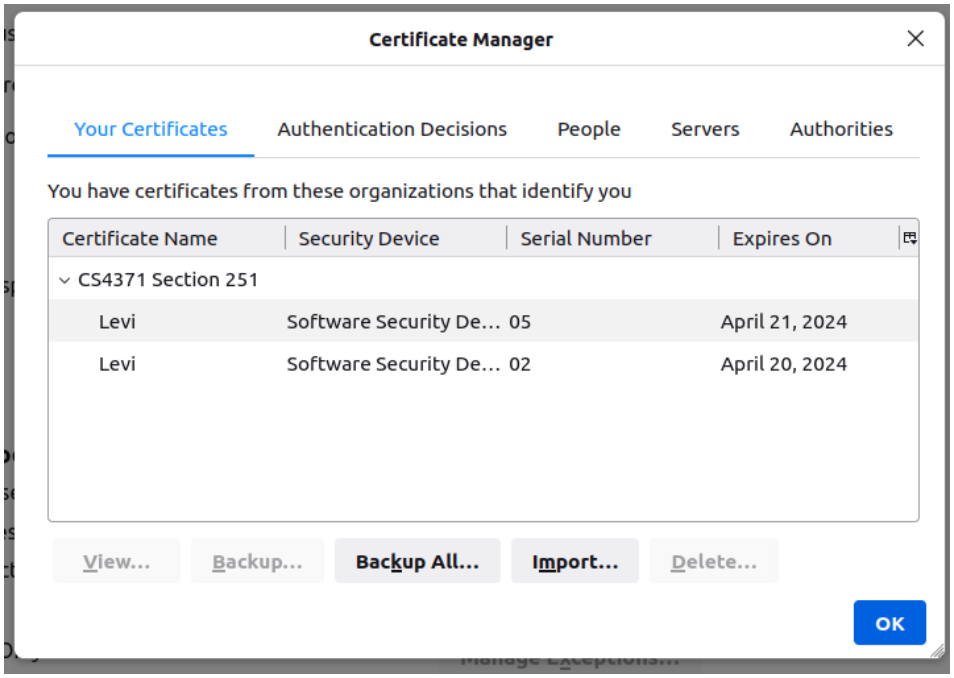
Creating the TLS client certificate with the issuing CA

```
leviadmin@leviadmin-VirtualBox:~/ca$ openssl ca \
> -config issuing-ca.cnf \
> -out ~/certs/cs4371-client.csr \
> -out ~certs/cs4371-client.crt \
> -policy extern_pol \
> -extensions client_ext
Using configuration from issuing-ca.cnf
Enter pass phrase for ./ca/issuing-ca/private/issuing-ca.key:
leviadmin@leviadmin-VirtualBox:~/ca$
```

Signing the tls-client certificate

```
leviadmin@leviadmin-VirtualBox:~/certs$ openssl pkcs12 -export -inkey cs4371.client.key -in cs4371-client.crt -out TLS-Client.p12
Enter pass phrase for cs4371.client.key:
Enter Export Password:
Verifying - Enter Export Password:
```

Creating the .p12 file for the tls-client



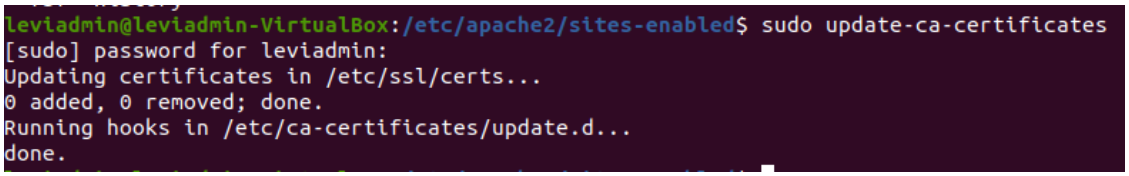
Importing the TLS client .p12 bundle into Firefox

```
#   A self-signed (snakeoil) certificate can be created by installing
#   the ssl-cert package. See
#   /usr/share/doc/apache2/README.Debian.gz for more info.
#   If both key and certificate are stored in the same file, only the
#   SSLCertificateFile directive is needed.
SSLCertificateFile      /home/leviadmin/certs/ca4371.local.crt
SSLCertificateKeyFile /home/leviadmin/certs/caD4371.local.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#                to point to the certificate files. Use the provided
#                Makefile to update the hash symlinks after changes.
SSLCACertificatePath /etc/ssl/certs/
SSLCACertificateFile /etc/ssl/certs/ca-certificates.crt
```

Editing the default-ssl.cnf to use the newly created certificates



Updating the certificates

Lessons Learned

- Security is incredibly important. Implementation of Certificate Authorities requires knowledge of what each CA does and the significance of each CA. We understand the dangers of having the Root CA on the live system and understand that it should be kept offline.
- A one-tier hierarchical design is unsafe due to a compromise of the one CA would mean a compromise of the entire PKI.

Future Recommendations

Future recommendations or improvements consist of an analysis of tiers needed, and level of security needed. For a real world example we would most likely include 3 tiers. As well as an implementation with hardware security modules (HSMs) that would aid in protecting private keys and cryptographic operations. We would potentially use a larger key size for the private keys used within the PKI.

Additional recommendations:

- Establish primary and backup sites (that are geographically separate).
- After hiring qualified individuals to make up our new legal department, we recommend that they be involved in all policy creation in case the PKI affects future customers or partners.
- Purchase a CA certificate from a commercial CA and issue certificates within the organization from internal, self-managed CAs that are connected to the external root CA.
- Develop a basic plan of action for compromise before a compromise occurs.

Appendix

Link to GitHub: .cnf files https://github.com/Brandon-Swanwick/two-tier-PKI