

Digital Certificates Introduction

Mike Sweeney
Packetattack

X509 and PKI Terminology

- **PKI (Public Key Infrastructure)**
 - Infrastructure using pairs of public and private keys to ensure privacy and security of data.
- **Private Key**
 - One half of the PKI key pair and which is never given out to anyone. This key can be protected by a password.
- **Public Key**
 - The second half of the PKI key pair that can be given out to the public or placed on a key server.
- **CSR (Certificate Signing Request)**
 - A request for a digital certificate.
- **CA (Certificate Authority)**
 - A trusted authority that signs your certificates and validates the certificate. This can be your own or public like Verisign, Thawte and others.

Terminology continued

- **CRL (Certificate Revocation List)**
 - A digitally signed list of revoked certificates that should not be honored any more.
- **Self Signed Certificate**
 - A digital certificate that you signed and validated using a private CA and is not trusted by default in the real world.
- **SSL (Secure Socket Layer)**
 - A protocol developed by Netscape to provide a secure method of exchanging data using a browser over a public network such as the internet.
- **Hash**
 - Hashing is taking a file and applying a hashing algorithm to it and generating a mathematical checksum. Now even if a single bit is changed, the checksum will not match and the file is now suspect.

Terminology Continued 3

- **RSA (Rivest, Shamir and Aldeman)**
 - A public key cipher that can be used to encrypt and to digitally sign.
- **DSA (Digital Signing Algorithm)**
 - Like RSA, it is a public key cipher but it can only be used to digitally sign.
- **PKCS#12 (Personal Information Exchange Syntax)**
 - A portable container file format that can be used to transfer certificates and private keys.
- **AES (Advanced Encryption Standard)**
 - A symmetrical cipher intended to replace DES
- **IDEA (International Data Encryption Algorithm)**
 - A drop in replacement cipher for DES
- **DES and 3DES (Data Encryption Standard)**
 - Cipher first used in 1976

Terminology Continued 4

- MD5
 - Message Digest 5. Hash algorithm that was released in 1991.
- SHA1
 - Secure Hash Algorithm version 1 cipher was released in 1995.
There are improved versions of SHA collectively called “SHA2”
-

A very Brief Cryptography History

- Cryptography is the art of encrypting and decrypting information
- The earliest known use of cryptography was the Egyptians in 1900BC
- Julius Ceaser use a simple substitution cipher to secure his communications
- Thomas Jefferson, in 1790, invented a cipher wheel that the US Navy used in WWII
- In the late 20s and early 30s, the FBI established an office to combat the increasing use of cryptography by criminals (rum runners)
- Dr. Hosrt Feistel invented the precursor to DES while at IBM
- In 1976, the Federal Govt introduced DES based on using Feistel ciphers
- In 1977, Scientific American magazine introduced us to Rivest, Shamir and Aldeman or RSA encryption. They offered it for free with a self addressed envelope and the NSA promptly freaked out
- 1990 brought us the 128 bit cipher called "IDEA"
- X.500 was the original protocol in 1988
- X.509 V3 was released in 1996

Uses of X.509

Common uses of X.509 digital certificates

- 802.1x port authentication
- Digital Signatures
- File Encryption
- Web Authorization (SSL)
- IP Security (end-point to end-point)
- Secure Email
- VPNs

Operating System Support

- Virtually any version of Linux using OpenSSL from CLI or alternative GUI tools such as TinyCA or SimpleCA
- Supported natively in Apple's OSX in Aqua or from the CLI.
- Supported in Windows 2003 server with the additional installation of IIS and the CA plugin.
- BSD uses OpenSSL to generate keys and manage certs
- Solaris supports X.509 using the “keytool” utility

Steps to creating a certificate

- Step one is to generate the private key
 - /usr/local/ssl/bin
 - **openssl genrsa -des3 -out mykeys.key 2048**
- Step two is to create a CSR
 - **openssl req -new -key mykeys.key -out request.csr**
- Step three is self-sign the certificate (optional)
 - **openssl req -new -key mykeys.key -x509 -out mycertificate.crt**

Step One details

- RSA Key can be generated using des, des3 (Triple DES) or using IDEA.
- Modulus can be 768, 1024 or 2048 bits in length but the default is 512.
- Recommended length is 2048 bits.
- More details on openssl can be found at: <http://www.openssl.org/>

```
MAC-G5:~ mikesweeney$ openssl genrsa -out applekey.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
MAC-G5:~ mikesweeney$
```

```
MAC-G5:~ mikesweeney$ less applekey.key
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAY3Qq+YSZ9E5nXGnk9+zC0k8EXBG7ApdcOBx/
/fxUMh1DmBZ5a4kOSkLRkh+2Bz2BIFNVA+d35VHzTQtgRy/UgJDFS
39hp2b5i5FHvLnuGcTJ9eoJU7IDplwe1sjwDIUrR/yaMXjLnw56u6hzbO
FBtY2qaZtjnsUma8/pABS/WyBT3LEb2TZOEtFRm/0y4e27Q7RKUC2G
qeHzUAKjNqcexMxlCWDBN2mNZsmAkPF/g/CyQH4y6NwUT2wmOdiT
/DzzS4ZGuEIPr9NuWaUqsgwf3c59Jv0UW/4TSQIDAQABAoIBAAFY6Z
VZzWsouV/I9ltqITu0ab2bOnsbzsfDKFR3+XFi+mxfXR4Ef4BmlzgxPGpf
leL+u2qHWmceNisChBO94bCXH0IplfXjXC0XT8wjWKBOcPKNF+3h+
qEBf6V4puq5H4RIWoDZJbLscKc9+P6ikCaHVhtD6gGX8CsBbmAmkD
qwwu7my9iZdydZw1UgB7poQ9FdljFsbTAOEbTucoGvlHixtQbXRhapk
35pju+6MtfnairJalAbD166aMJQnSaiQhWDjt7O2UKFJ5H1b2s8IRq0bX
SBQmX8kCgYEA+9I1id4WZnI9DwNjGeU0DY34ARax6zKQ1WRY8KU
9iW1PWPkIrkp7GBkepCxAej6AsgmUXDJ4IjzldZHxGeH6AYgRpDukyJ
M+59omaSIAHuwDZI/zaCiWkJN0h8Fs9gJdKVLHQ882TXpjc6W/XScz
mtSZNVqob0c5fBl1aRxi3IXSctM5/CRY7i5kh4qxITh2SF8lwZeE2Hgf
eYEBaoGAFMhDFA/mxScAJoa1iHiOXW2sAKjBW2UcDnoDIQEDcQS
8Mi2VJe5kjjYP5co2G0r+hmX4uDr20V2P8f/wxx24xcMmiPFeth3br/7cX
mTlzT1BvxbC37wjCWIQgpRxfY8bZSdK8f+q38Hp69oVHptlq9o=
-----END RSA PRIVATE KEY-----
```

Step Two Details

- We need to create a CSR using the key that we just created.
- The CSR will be either sent to a trusted source or self signed.

```
MAC-G5:~ mikesweeney$ openssl req -new -key  
applekey.key -out appleG5.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) [AU]:**US**

State or Province Name (full name) [Some-State]:**California**

Locality Name (eg, city) []:**Orange**

Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Packetattack

Organizational Unit Name (eg, section) []:**Engineering**

Common Name (eg, YOUR name) []:**Mike Sweeney**

Email Address []:**mikesweeney@packetattack.com**

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []:

An optional company name []:

```
MAC-G5:~ mikesweeney$
```

Resulting appleG5.csr file

-----BEGIN CERTIFICATE REQUEST-----

```
MIIC6jCCAdICAQAwwgaQxCzAJBgNVBAYTAIVTMRMwEQYDVQQIEwpDYWxpZm9ybmlh
MQ8wDQYDVQQHEwZPcmFuZ2UxFTATBgNVBAoTDFBhY2tldGF0dGFjazEUMBIGA1UE
CxMLRW5naW5lZXJpbmcmFTATBgNVBAMTDE1pa2UgU3dlZW5leTErMCKGCSqGSIb3
DQEJARYcbWlrZXN3ZWVuZXIAcGFja2V0YXR0YWNrLmNvbTCCASlwdQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAMt0KvmEmfROZ1xp5PfswtJPBFwRuwKXXDgcfx5R
CvtqnYAPWP38VDIdQ5gWeWuJDkpC0Ziftgc9gZRTVQPnd+VR800LYEcv1ICQxUrN
LsRVAzqV3N/Yadm+YuRR7y57hnEyfXqCVOyA6ZcHtbI8A5VK0f8mjF4y58Oeruoc
2znZLkcSihQbWNqmmbY57FJmvP6QAUv1sgU9yxG9k2ThLX0Zv9MuHtu0O0SIatho
pKUYBKuWDKnh81ACozanHsTMSHMAwTdpjWbJgJDxf4PwskB+MujcFE9sJjnYkzeb
tqxERWgXZfw880uGRrhJT6/TblmlKrlMH93OfSb9FFv+E0kCAwEAAaAAMA0GCSqG
SIb3DQEBAUAA4IBAQCcrJi0cUSw7LsDyoWKLjM9Tq4MSep5zqONtya9UB4PpqUnc
NTcVBNUQkxt03Oz6q1HGLcnPsRAzxNRDA7OkMg/b8bV3uYaLC1NuyKkur0k0UVyh
5pVW5juXy5iROUhBvz7iJNPYutA23YWtfHqyOXJ08/pu6Y8xS4NwTFm0WWW/h4aG
4ETnrjXaTAAdVqeioTm7LzYX59rF4JIQpvUJKIYHfwl+ZRiuC8vPIBDuE0vx44ik
PfGcf6OoVRI7L/CpnyiXJDZo/wfRaLZL5a0nWtm/1XYq5xF3oqTfSW9wZrWRhTte
P/Pp8X36DXEZcZFDB7gvT8+rpdqeWjM3MHAABGV0
-----END CERTIFICATE REQUEST-----
```

Optional Step three details

- We can self-sign our certificate request using openssl.
- We can use the self-signed certificate for any use we might need so long as we understand that this certificate while trusted for internal use is not trusted outside of our network.

```
MAC-G5:~ mikesweeney$ openssl x509 -req -days 720 -in appleG5.csr
-signkey applekey.key -out mycertificate.crt
Signature ok
subject=/C=US/ST=California/L=Orange/O=Packetattack/OU=Engineering
/CN=Mike Sweeney/emailAddress=mikesweeney@packetattack.com
Getting Private key
MAC-G5:~ mikesweeney$
MAC-G5:~ mikesweeney$ less mycertificate.crt
-----BEGIN CERTIFICATE-----
FBhY2tldGF0dGFjazEUMBIGA1UECxmLRW5naW5lZXJpbmcmxFTAT
BgNVBAMTDE1pa2UgU3dlZW5leTErMCKGCSqGSIb3DQEJARYcbW
cGFja2V0YXR0YWNrLmNvbTCCASlwdQYJKoZIhvcNAQEBBQADgg
AMt0KvmEmfROZ1xp5PfswtJPBFwRuwKXXDgcfx5RCvtqnYAPWP38
DkpC0Zlftgc9gZRTVQPnd+VR800LYEcv1ICQxUrNLsRVAzqV3N/Yau
hnEyfXqCVOyA6ZcHtBl8A5VK0f8mjF4y58Oeruoc2znZLkcSihQbWN
vP6QAUv1sgU9yxG9k2ThLX0Zv9MuHtu0O0SIathopKUYBKuWdKnh
SHMAwTdpjWbJgJDxf4PwskB+MujcFE9sJjnYkzebtqxERWgXZfw880
blmIKrIMH93OfSb9FFv+E0kCAwEAATANBgkqhkiG9w0BAQQFAAO
NmNsBg/ZpU9L1Mi4a1ROcqM1HcgYdiU5xPb3ipv9SgWkxQcK87bm
VAPBWT0npg0HHgO7pDBTqLAZ1j6gmt/Q6XAmbQLPpe9ERzJ0f5H
22AzRTx5sdJsMTL5wmaqtlv9qN+ajn1Bku4e9vgeTAzjdWx5GHBRaZ
JGHD/Vrg0WROdfofu2UltGvDdnxpBWVJW4lzn0dDa0VFAf9xUh7v5rl
qqI+9rWv/Z3c6BsD3t/Mdt6qCWse85k+LmyBTf9uu8cjdk4945yqN/qpR
4gz7R6XtLzCc7Q==
-----END CERTIFICATE-----
mycertificate.crt (END)
```

OpenSSL CA.pl

- We can use a script included with OpenSSL called CA.pl which gives a “friendlier way” to make our certificates
- We start with using the “-newreq” command.
- The “newreq” parameter starts the process of making a new certificate request and keys.
- The resulting file is called “newreq.pem”

```
MAC-G5:~ mikesweeney$ /System/Library/OpenSSL/misc/CA.pl -newreq
Generating a 1024 bit RSA private key
....+++++
.....+++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Orange
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Packetattack
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:Mike Sweeney
Email Address []:mikesweeney@packetattack.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
MAC-G5:~ mikesweeney$
```

OpenSSL CA.pl

- We now use the “-newca” command
- The “newca” parameter starts the process of making a new certificate authority to self sign our new request.
- The results are a new directory structure and files are created the first time this command is used.
- The default directory is “demoCA”.

```
MAC-G5:~ mikesweeney$ /System/Library/OpenSSL/misc/CA.pl -newca  
CA certificate filename (or enter to create)
```

```
Making CA certificate ...
```

```
Generating a 1024 bit RSA private key
```

```
.....++++++
```

```
.....++++++
```

```
writing new private key to './demoCA/private/cakey.pem'
```

```
Enter PEM pass phrase:
```

```
Verifying - Enter PEM pass phrase:
```

```
phrase is too short, needs to be at least 4 chars
```

```
Enter PEM pass phrase:
```

```
Verifying - Enter PEM pass phrase:
```

```
-----
```

```
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) [AU]:US
```

```
State or Province Name (full name) [Some-State]:California
```

```
Locality Name (eg, city) []:Orange
```

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Packetattack.com
```

```
Organizational Unit Name (eg, section) []:Engineering
```

```
Common Name (eg, YOUR name) []:Mike Sweeney
```

```
Email Address []:mikesweeney@packetattack.com
```

```
MAC-G5:~ mikesweeney$ cd demoCA
```

```
MAC-G5:~/demoCA mikesweeney$ ls
```

```
cacert.pem    crl          newcerts     serial
```

```
certs        index.txt   private
```

```
MAC-G5:~/demoCA mikesweeney$
```

OpenSSL CA.pl

- Lastly, we use the “-signreq command.
- The “signreq” parameter will self sign our certificate and assign some defaults such as life span of certificate (365 days).
- The output file is called “newcrt.pem”

```
MAC-G5:~ mikesweeney$ /System/Library/OpenSSL/misc/CA.pl -signreq
```

```
Using configuration from /System/Library/OpenSSL/openssl.cnf
```

```
Enter pass phrase for ./demoCA/private/akey.pem:
```

```
Check that the request matches the signature
```

```
Signature ok
```

```
Certificate Details:
```

```
Serial Number:
```

```
e2:d9:e1:cd:b9:77:d5:8c
```

```
Validity
```

```
Not Before: Nov 22 06:58:33 2005 GMT
```

```
Not After : Nov 22 06:58:33 2006 GMT
```

```
Subject:
```

```
countryName          = US
```

```
stateOrProvinceName  = California
```

```
localityName         = Orange
```

```
organizationName     = Packetattack
```

```
organizationalUnitName = Engineering
```

```
commonName           = Mike Sweeney
```

```
emailAddress         = mikesweeney@packetattack.com
```

```
X509v3 extensions:
```

```
X509v3 Basic Constraints:
```

```
CA:FALSE
```

```
OpenSSL Generated Certificate
```

```
X509v3 Subject Key Identifier:
```

```
CE:F7:50:6A:3D:EE:12:B7:EC:9C:3C:FE:A0:36:E8:52:5E:AD:B5:38
```

```
X509v3 Authority Key Identifier:
```

```
keyid:2C:E0:C3:0D:8E:60:12:9E:27:11:32:0B:B8:47:BD:30:B9:26:62:3F
```

```
DirName:/C=US/ST=California/L=Orange/O=Packetattack.com/OU=Engineering/CN=Mike
```

```
Sweeney/emailAddress=mikesweeney@packetattack.com
```

```
serial:E2:D9:E1:CD:B9:77:D5:8B
```

```
Certificate is to be certified until Nov 22 06:58:33 2006 GMT (365 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
```

```
Signed certificate is in newcert.pem
```

```
MAC-G5:~ mikesweeney$
```

Other CA.pl Commands

- newreq-nodes
 - Works like -newreq but the private key is not encrypted
- pkcs12
 - This creates a PKCS#12 container file with the user certificate, private key and CA certificate.
- signcert
 - This is the same as -sign but it expects a self-signed certificate to be in the “newreq.pem” file.
- verify
 - This will verify certificates against the CA certificate for “demoCA”

X.509 Cost

- Trusted certificates can be expensive but there is not any further work required on the client side.
- Self signed certificates are free but require the importation of the certificate and the CRL on each client.
- There is a free service of a trusted CA called Cacert.org
- OpenSSL can generate certificates and is available to compile on many different platforms or is included already in some commercial operating systems such as OSX
- Windows 2003 server can also generate certificates with the downloadable application.

How SSL uses X.509

How SSL works using x.509

- 1: Alice contacts Bob and requests a SSL session to be opened
- 2: Bob sends Alice his digital certificate and public key
- 3: Alice compares the certificate to the trusted certificate database. If the certificate is not listed then it needs to be added.
- 4: Alice uses Bob's public key to encrypt a session key and sends it to Bob

How SSL uses X.509 part 2

- 5: Bob gets the encrypted session key and can ask for a certificate also at this time but assuming a certificate is not asked for, Bob will decrypt the session key using his private key
- 6: If the session key decrypts properly, Bob will send an acknowledgement to Alice and open a secure data channel to Alice.

PKI Security Risks

- Who do we trust?
 - Who gave the CA the authority to be “trusted”? There is a single meaning of “trust” in the cryptographic world, it means that they control their own private key.
- Who is using my private key?
 - One of the biggest risks is the control over your private key. The vendors have borrowed a term from the cryptographic world called “non-repudiation” which means specifically that the cipher is unbreakable. The vendors use this term legally to say you can not challenge that someone or something else used your digital signature.
- How Secure is the verifying computer system?
 - Not only does your system need to be secure but the host on the other end needs to be trusted. But, an attacker could issue his public key for a fake root certificate and now he is “trusted”

PKI Security Risks 2

- Which Bob Smith is this?
 - We trust based on the names in the certificate but what if there is a Bob Smith that steals a 2nd Bob Smith's key set and uses it? Now using names is not such a good idea is it?
- Is the CA an authority?
 - Just because the CA can issue certificates does not make them an authority on the contents. An example is DNS. There are authorities to control DNS naming and issuing of the names but none of the CAs that issue SSL certificates based on DNS names are one of these authorities.
- Is the user part of the security design?
 - SSL security can not react to nor can it control the content of a web page. All the SSL certificate knows about is the DNS address which may not even be the business name depending on who is hosting. The user can not be expected to sort all of this out if we can cant.

PKI Security Risks 3

- Was the certificate issued by a single CA or by a CA and registration authority?
 - Any chain is only as strong as its weakest link and the CA-RA certificate is the weakest. The CA could forge the documents but they normally promise not to. But humans are involved and there is the weakness.
- How did the CA identify the certificate holder?
 - Some CAs do not consider who is really holding the private key to be part of the application process and to something to be verified in some manner.
- How secure are the certificate security practices?
 - Some of the current “security practices” that the CAs use were just arbitrary points picked in the early days without a lot of thought to them. Or it was fast such as using 512 bit keys instead of 2048 bit keys. Does the CA support using CRLs? Not all do.
 - Much of these last three slides came from a paper from Bruce Schneier who is a world renowned cryptographic expert and the author of the Blow Fish cipher.

Resources

- Trusted CAs
 - [Verisign, Thawte, Cacert.org](#)
- X.509 Certificate and CRL Profile
 - <http://www.ietf.org/rfc/rfc3280.txt>
- Peter Gutmann's Certificate Style Guide
 - <http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt>
- The National Cryptologic Museum
 - <http://www.nsa.gov/museum/index.cfm>
- RSA Security
 - <http://www.rsasecurity.com/>
- PKCS
 - <http://www.rsasecurity.com/rsalabs/node.asp?id=2138>
- Security Risks of PKI
 - <http://www.schneier.com/paper-pki-ft.txt>