Encryption, Certificates and SSL

DAVID COCHRANE
PRESENTATION TO BELFAST OWASP CHAPTER
OCTOBER 2018
Agenda

• Basic Theory: encryption and hashing
• Digital Certificates
• Tools for Digital Certificates
• Design Patterns
• Case Study – Build your own CA – Slides not published
Encryption Basics

• AES
• Blowfish
• Twofish
• 3DES
• RC2, RC4

• RSA
• Elliptic Curve
Cryptographic Hashing

• Converts a large amount of data to a “representative” number
  • MD5 – 128 bits
  • SHA1 – 160 bits
  • SHA2 / SHA256 – 256 bits
  • SHA384, SHA512 – 384, 512 bits respectively

• One-way process

• Used to verify that two files or strings are the same — without checking both byte by byte

• Safe storage of passwords

• Importance of “Salt”

• Digital signature = EncryptUsingPrivateKey (Hash(Data))
What’s a Digital Certificate?

- Data that represents an entity or object and can be used to verify its identity
- Attributes are defined by X.509

Issued to: CN=David Cochrane,O=Viridian Group,C=GB,L=Belfast
Issued by: CN=Certifying Authority,O=Your favourite CA,…

- Subject and issuer (X.500 format)
- Subject public key
- Start and end dates
- Serial number and hash
- Usage (Basic and enhanced)
- Alternate names (DNS or IP)
- Subject key ID and authority key ID
- Revocation (CRL and/or OCSP)
- Digital signature of issuer
- …
What Makes a Certificate Trusted – PKI

- Certificate Chain
  - Certifying Authority (CA)
  - Intermediate Certificates
- Start Date and Expiry
- Revocation
  - CRL or OCSP
Windows and Digital Certificates

• **Windows Certificates Stores**
  - Two physical stores: User and Machine
  - Logical stores in each: Personal, Trusted Root Certification Authorities, Other People, Trusted People, Trusted Publishers, Intermediate Certification Authorities, Active Directory User Object, …

• **Windows Keystores**
  - Private key is stored in a keystore separately from the certificate
  - Separate keystore for each Crypto service provider, e.g.
    - Microsoft Enhanced RSA and AES Cryptographic Provider
    - Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
  - User keystores protected by user key, which is derived from user’s password
  - Machine keystores protected by machine key – needs local admin to access
  - Private key memory is protected by Windows Crypto system, smart card or TPM chip
Digital Certificate Formats

- DER – binary encoded using ASN.1 (Windows file type .cer)
- PEM – Base64 encoded, separate KEY file contains encrypted key or encrypted within the PEM file
- File type .crt can be either a DER or PEM file
- PKCS12 store (PFX or P12) – contains private key protected by password, can contain multiple certificates, e.g., complete certificate chain
- PKCS7 store – contains multiple certificates similar to DER (file type .p7b)
- CSR – certificate signing request, no key details or digital signature
- JKS – Java keystore for users of Sun’s Java crypto library
Certificate Signing Requests

• Details of the certificate to be signed, similar to CER format
• Private key is stored on the server that generated the request
• Usually uploaded to Certifying Authority’s web site so that signed certificate can be downloaded
• Certificate signing requests can be generated by IIS, Windows certificate manager or OpenSSL
  • Key length, usage, algorithm, alternate names, …
• Private key is matched with the signed certificate when it is installed on the server
Certificate Attributes – OpenSSL

Data:

Version: 3 (0x2)
Signature Algorithm: sha512WithRSAEncryption

Issuer: OU = Technology and Change Team, O = Viridian Group Limited, CN = Viridian Group Certifying Authority

Validity

Not Before: Oct 10 15:19:12 2016 GMT
Not After : Oct 10 15:19:12 2019 GMT

Subject: CN = David Cochrane, emailAddress = David.Cochrane@viridiangroup.co.uk

Subject Public Key Info:

Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)

X509v3 extensions:

X509v3 Extended Key Usage: critical
TLS Web Client Authentication, E-mail Protection, Code Signing

X509v3 Subject Key Identifier:


Signature Algorithm: sha256WithRSAEncryption
Certificates and SSL/TLS

- Server certificate provided during negotiation must be trusted by the client browser. Firefox, Safari use their own certificate stores.
- Wildcard and Subject Alternate Name certificates allow one certificate for multiple sites.
- Certificates for an Internet site can be requested from public CA via a CSR.
  - Can’t use internal server name or IP address.
- Domain Validated certificates vs Extended Validation certificates.
- One year up to five years validity.
- https://ssllabs.com/ssltest
Certificates and SSL/TLS

• Make sure you specify enough X.500 attributes: CN, C, O, (OU), (L)
• Allow for all of the possible names as Subject Alternate Names or use a wildcard certificate
• Specify RSA 2048 or better for encryption and SHA256 or better for hashing
• Alternatively, ECC 256 bit is acceptable
• Specify Microsoft Enhanced Crypto Provider to store the private key
• For IIS install the certificate into the Machine Personal store (or install using IIS Server Certificates option)
• Update config file for web servers other than IIS: Apache, Tomcat, Weblogic, …
Sample Certificate - Wikipedia

Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 08:30:94:62:d1:fed6:0a0e:0baf:15:af88:bc5:45
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
Validity
  Not Before: Dec 21 00:00:00 2017 GMT
  Not After: Jan 24 12:00:00 2019 GMT
Subject: C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation, Inc.", CN = *.wikipedia.org
Subject Public Key Info:
  Public Key Algorithm: id-ecPublicKey
  Public Key: (256 bit)
  ASN1 OID: prime256v1
  NIST CURVE: P-256
X509v3 extensions:
  X509v3 Authority Key Identifier:
  X509v3 Subject Key Identifier:
  X509v3 Subject Alternative Name:
    DNS:*.wikipedia.org,
    DNS:*.m.wikipedia.org,
    DNS:*.wikimedia.org,
    DNS:*.wikibooks.org,
    DNS:*.wikdata.org,
    DNS:*.wikicite.org,
    DNS:*.wikiversity.org,
    DNS:*.wikinews.org,
    DNS:*.wikiversity.org,
    DNS:*.wikisource.org,
    DNS:*.wikivoyage.org,
    DNS:*.wiktionary.org,
    DNS:*.wikiversity.org
X509v3 Key Usage: critical
  Digital Signature
X509v3 Extended Key Usage:
  TLS Web Server Authentication, TLS Web Client Authentication
Sample Certificate - Wikipedia

X509v3 CRL Distribution Points:
Full Name:
URI:http://ca3.digicert.com/sha2-2ha-server-g6.crl
Full Name:
URI:http://ca4.digicert.com/sha2-2ha-server-g6.crl

X509v3 Certificate Policies:
Policy: 2.16.840.1.114412.1.1
CPS: https://www.digicert.com/CPS
Policy: 2.23.140.1.2.2

Authority Information Access:
OCSP - URI:http://ocsp.digicert.com

X509v3 Basic Constraints: critical
CA:FALSE

CT Precertificate SCTs:
Signed Certificate Timestamp:
Version : v1 (0x0)
Timestamp : Dec 21 18:11:19.631 2017 GMT
Extensions: none
Signature : ecdsa-with-SHA256

Signed Certificate Timestamp:
Version : v1 (0x0)
Timestamp : Dec 21 18:11:19.720 2017 GMT
Extensions: none
Signature : ecdsa-with-SHA256

Signature Algorithm: sha256WithRSAEncryption
Important Tools for Digital Certificates

• **Windows**
  - Certificate Manager (MMC or certmgr.msc) – GUI for managing certificates in Windows
  - Certificate Utility – Windows certificate services
  - MAKECERT – Basic tool to create certificates, part of Windows SDK
  - SignTool – Code signing, part of Windows SDK
  - Encrypted File System – the easy way to encrypt files
  - ASPNET_REGIIS – encrypts / decrypts .Net web config files

• **Cross-platform**
  - OpenSSL – powerful command line tool to do almost anything with certificates
  - Keytool – creates and modifies JKS files (part of Java Developer Kit)
OpenSSL Commands

X509 – display and convert DER and PEM certificates
PKCS12 – create, verify and display PFX files
REQ – create and display certificate requests
OCSP – check certificate validity using OCSP
GENRSA – create an RSA key
ENC – encrypt or decrypt
CA – functions to act as a basic certifying authority
S_CLIENT – make an SSL / TLS connection to a web site, FTPS server or SMTP server
Libraries

- Windows CryptoAPI (some functions now deprecated W10 / WS2016)
- Windows CNG (Cryptography API Next Generation) – ECC support
- OpenSSL
- Java Cryptography Architecture
- Bouncy Castle
- …
Design Patterns

- .Net web site – storing application passwords securely
- Windows application – storing passwords or SSH keys securely
- Encrypt a file using a certificate
- Using a certificate for web site authentication
- Validate user login and password
- Verify user identity in a client application and an Intranet site
.Net web site – storing passwords securely

• **Method 1** (recommended)
  - Store password in web config file and use ASPNET_REGIIS as follows:
    - `ASPNET_REGIIS -pe "PasswordSection" Webroot` (to encrypt)
    - `ASPNET_REGIIS -pd "PasswordSection" Webroot` (to decrypt)

• **Method 2**
  - Store the password in a separate file that has been manually encrypted using certificate
  - Install the certificate in the machine certificate store
  - In your application code load the certificate, and use private key to decrypt the contents of the file
Storing passwords or SSH keys securely

• **Method 1** (recommended for server applications)
  - Use Windows EFS to encrypt the file using the credentials of the account the application will run under

• **Methods for client applications**
  - Use a secure web service to retrieve the password or SSH key
    - Configure the web service to run under the user’s credentials then retrieve those
  - OR Use an encrypted Kerberos connection to a server-based application (see later) to retrieve password or key
Encrypt a File using a Certificate – Method 1

- Access certificate from store
- Obtain public key
- Read file contents
- Encrypt using public key
- Save file contents

```csharp
X509Store store = new X509Store("My");
X509Certificate2Collection collection = store.Certificates;
X509Certificate2 certificate = collection.Find(FindBySubjectDistinguishedName, "Encryptor")[0];
RSACryptoServiceProvider encryptor = certificate.PublicKey;

// Read file contents into byte[] clearData
encryptor.Encrypt(clearData, encryptedData);

// Overwrite file with encrypted data
```
Encrypt a File using a Certificate – Method 2

- Create a random symmetric key
- Access certificate from store
- Obtain public key
- Read file contents
- Encrypt symmetric key using public key
- Encrypt file using symmetric key
- Save encrypted symmetric key and encrypted file contents

```csharp
AESCryptoServiceProvider aes = new AESCryptoServiceProvider();
aes.GenerateKey();
X509Store store = new X509Store("My");
X509Certificate2Collection = store.Certificates;
X509Certificate2 certificate = collection.Find(FindBySubjectDistinguishedName, "Encryptor")[0];
RSACryptoServiceProvider encryptor = certificate.PublicKey;
// Read file contents
encryptor.Encrypt(aes.Key, encryptedKey);
ICryptoTransform aesEncrypt = aes.CreateEncryptor(aes.Key, aes.IV);
// Encrypt data a block at a time using aesEncrypt
// Overwrite file with encrypted key and data
```
Using a Certificate for Authentication

**CLIENT**

WebRequestHandler handler = new WebRequestHandler();

X509Certificate certificate = GetClientCertFromStore();

handler.Client Certificates.Add(certificate);

HttpClient client = new HttpClient(handler);

HttpResponse response = await client.GetAsync(URL);

**SERVER**

![SSL Settings](image-url)

This page lets you modify the SSL settings for the content of a Web site or application.

- **Require SSL**
- **Ignore**
- **Accept**
- **Require**
Validate User Login and Password – Method 1

- Retrieve user’s hashed password
- Hash supplied password
- Compare them

bool function IsPasswordValid (string username, string enteredPassword)
{
    // SELECT UserHash FROM Users WHERE Login = ?username
    SHA256CryptoServiceProvider hasher = new SHA256CryptoServiceProvider();
    hasher(Encoding.UTF8.GetBytes(enteredPassword + salt), enteredHash);
    for (int i = 0; i < userHash.ArraySize; i++)
        if (enteredHash[i] != userHash[i]) return false;
    return true;
}
Validate User Login and Password – Method 2

• Retrieve user’s encrypted password
• Decrypt it
• Compare with entered password

• What are the two coding flaws?

```csharp
bool function IsPasswordValid(string username, string enteredPassword)
{
    // SELECT encryptedPassword FROM Users WHERE Login = ?username
    key.Decrypt(encryptedPassword, clearPasswordBytes);
    string clearPassword = Encoding.UTF8.GetString(clearPasswordBytes);
    return (clearPassword == enteredPassword);
}
```
Verify User Identity

Client Application

```csharp
WindowsIdentity wi = WindowsIdentity.GetCurrent();
WindowsPrincipal wp = new WindowsPrincipal(wi);
string username = wp.Identity.Name;
```

Intranet Web Application

```csharp
// Ensure IIS is configured to use Windows authentication and the
// web application pool runs under the user's identity
string username = Page.User.Identity.Name;
```
Questions