Secure Communication on the Web

CS-576 Systems Security
Instructor: Georgios Portokalidis
Fall 2018
Overview

Establishing encrypted connections using PK encryption
Passive vs active adversaries

Securing communications
- Message integrity
- Key authentication

TLS/SSL
Certificates and certificate authorities
Attacks against SSL/TLS
Establishing Encrypted Connections

Mutually agreed secret/session key

Encrypted session
Types of Adversaries/Attacks

**Passive** – does not affect system resources
- Can intercept messages but not modify

**Active** – attempt to alter system resources or affect their operation
- Can intercept, re-order, and alter messages
Passive Attacker

I can see what you exchange

Mutually agreed secret/session key

Encrypted session
Passive Attacker

Communication remains secure

Bob

PU_Bob

PR_Bob

Mutually agreed secret/session key

Alice

PU_Alice

PR_Alice

Encrypted session

I can see what you exchange
Active Attacker

Bob

PU_{Bob}

PR_{Bob}

Alice

PU_{Alice}

PR_{Alice}

Mutually agreed secret/session key

Encrypted session

I can intercept and modify your messages
Active Attacker

Communication is compromised

Mutually agreed secret/session key

Encrypted session

I can intercept and modify your messages

PU_{Bob} →\hspace{1cm}PU_{Alice}

Bob

PU_{Bob} \hspace{1cm} PR_{Bob}

Alice

PU_{Alice} \hspace{1cm} PR_{Alice}

Fall 2018

Stevens Institute of Technology
Alteration of Messages

Mutually agreed secret/session key

PU_{Alice}  m  PU_{Bob}

m'  m

PU_{Alice}  PR_{Alice}  PU_{Bob}  PR_{Bob}

Bob  Darth Vader  Alice
Alteration of Messages

Mutually agreed secret/session key

Message may be garbage after decryption
Message Integrity with MAC

Encrypted data need to be protected with MAC against active adversaries.

**MAC-and-Encrypt**

\[ E(P) || M(P) \]

- No integrity of the ciphertext

**MAC-then-Encrypt**

\[ E(P || M(P)) \]

- No integrity of the ciphertext

**Encrypt-then-MAC**

\[ E(P) || M(E(P)) \]

- The right option
Alteration of Messages Detected

Mutually agreed secret/session key

This message is not authentic
Man-in-the-middle (MITM)

Waiting for Bob and Alice to start talking
Man-in-the-middle (MITM)

Fully compromised channel

Mutually agreed secret/session key

PU_{Bob} → PU_{Darth} → PU_{Alice}

PU_{Darth} ← PU_{Bob} ← PU_{Alice}

Mutually agreed secret/session key

Bob

PU_{Bob} \quad PR_{Bob}

Darth

PU_{Darth} \quad PR_{Darth}

Alice

PU_{Alice} \quad PR_{Alice}
Public-Key Authenticity

PK encryption requires that parties can establish the authenticity of public keys.

Some ways to accomplish this:
- Trust on first use (TOFU)
- Web of Trust
- Public-key infrastructure (PKI)
Certificates

Certificates are essentially signed public keys
  - Signed with the private key of a certificate authority
Trusted Certificate Authorities

Bob

CA

Trust

Alice

Trust

Not sure
Mutually agreed secret/session key

PU_{Bob}

CERT_{Alice}

CERT_{Chris}  PU_{Bob}  PR_{Bob}  

CERT_{Alice}  PU_{Alice}  PR_{Alice}
Certificates

Unsigned certificate: contains user ID, user's public key, as well as information concerning the CA

Recipient can verify signature by comparing hash code values

Generate hash code of unsigned certificate

Encrypt hash code with CA's private key to form signature

Create signed digital certificate

Decrypt signature with CA's public key to recover hash code

Use certificate to verify Bob's public key

Figure 2.7  Public-Key Certificate Use

Generate hash code of unsigned certificate

Encrypt hash code with CA's private key to form signature

Create signed digital certificate

Use certificate to verify Bob's public key

Recipient can verify signature by comparing hash code values
Certificate Chains

Trust anchors: Systems are preconfigured with a list of trusted certificates
- System-wide or application-based store
- More can be added: self-signed, organization certificates, MiTM certificates, etc.

Server provides a chain of certificates
Any CA can sign certificates for any domain
- The system is as secure as the weakest CA
Transport Layer Security (TLS) is the most widely used protocol for secure communications over TCP.

Succeeds the Secure Socket Layer (SSL)
- Plagued by various security issues

Used in HTTPS, IMAPS, SMTP, etc.
TLS Protocols

Handshake protocol

- Negotiate sessions keys
- Authenticate server and (optionally) client
TLS Handshake

Client

- ClientHello
- ClientKeyExchange
- ChangeCipherSpec
- Finished

Server

- ServerHello
- Certificate [optional]
- ServerKeyExchange
- ServerHelloDone
- ChangeCipherSpec
- Finished

GET /login HTTP/1.1\r\n
TLS Protocols

Handshake protocol
- Negotiate sessions keys
- Authenticate server and (optionally) client

Record protocol
- Exchange messages encrypted and MACed with established session key
- Compression before encryption
  - Don’t do it
- Extensible sub-protocols
  - For example, change the cipher suit used
TLS Records

Client

- ClientHello

Server

- ServerHello
- Certificate [optional]
- ServerKeyExchange
- ServerHelloDone
- ClientKeyExchange
- ChangeCipherSpec
- Finished

GET /login HTTP/1.1\r\n
ChangeCipherSpec
- Finished
Problems with CAs

CAs are businesses doing this for profit
- Certificates are expensive
- Self-signed certs cost nothing

Despite the warnings users tend to keep going

Now you can a cert for free
- https://letsencrypt.org/

Fall 2018
Stevens Institute of Technology
Problems with CAs

CAs issuing invalid certs

Google Security Blog
The latest news and insights from Google on security and safety on the Internet

Chrome’s Plan to Distrust Symantec Certificates
September 11, 2017

Posted by Devon O’Brien, Ryan Sleevi, Andrew Whalley, Chrome Security

This post is a broader announcement of plans already finalized on the blink-dev mailing list.

Update, 1/31/18: Post was updated to further clarify 13 month validity limitations
Problems with CAs

Misplaced “CA” keys

DUST UP —
23,000 HTTPS certificates axed after CEO emails private keys

Flap that goes public renews troubling questions about issuance of certificates.

DAN GOODIN - 3/1/2018, 8:36 AM
Problems with CAs

Why is this root cert in my browser?

---

NOTHING "DISHONEST"? —

Turkish government agency spoofed Google certificate “accidentally”

CA mistakenly gave Ankara’s transit authority even more authority.

SEAN GALLAGHER - 1/4/2013, 3:44 PM

Microsoft has released a security advisory concerning a fraudulent digital certificate for all Google domains apparently created by the Turkish government. The certificate, which was created by a subsidiary Certificate Authority issued to the transportation directorate of the city government of Ankara, could have been used to intercept SSL traffic as part of a "man in the middle" attack to spoof Google's encryption certificate and decrypt secure Web sessions to Google Plus and GMail.

According to a statement from the Turkish certificate authority Turktrust, the organization mistakenly issued two organizations subsidiary CA certificates in 2011—created during testing of Turktrust's certificate production system—instead of the standard SSL certificates they were supposed to receive. Subsidiary CA certificates give the holder the ability to issue SSL certificates with the original CA's authority.
Downgrade Attacks

Goal: force the use of a weak cipher suite

Possible because browsers voluntarily downgrade the protocol upon handshake failure
  - For interoperability reasons
  - Due to server bugs
  - Due to protocol weaknesses

Methods:
  - Close connections until retry with lower SSL/TLS version
  - Modify list of supported ciphers sent from the client
Downgrading TLS Connection

- ClientHello (TLS 1.1)
  - RST
  - ClientHello (TLS 1.0)
    - RST
      - ClientHello (SSL 3.0)
    - ✓

Bob

Alice
Downgrade Cipher Suite

SSL handshake (I only support 512-bit RSA)  SSL handshake (I only support 512-bit RSA)

Bob

PU_{Bob}  PR_{Bob}  PU_Alice

Alice

PU_{Bob}  PR_{Bob}  PU_Alice

Can be factored in less than 12 hours for $100 on Amazon EC2
Downgrade Cipher Suite

SSL handshake (I only support 512-bit RSA)

Bob

PU_{Bob}  PU_{Alice}  PU_{Bob}

https://www.smacktls.com/

Alice

PU_{Alice}  PU_{Bob}  PU_{Alice}

Can be factored in less than 12 hours for $100 on Amazon EC2
SSL Stripping

![Diagram showing SSL stripping attack]

- **Victim**
- **Attacker**
- **Server**

Location: `http://...`  
Location: `https://...`

`<a href="http://...">`  
`<a href="https://...">`

`<form action="http://...">`  
`<form action="https://...">`
HSTS

HTTP Strict Transport Security protects against SSL stripping and other attacks

- Convert any insecure links to https
- Treat all errors as fatal

Implemented through an HTTP header

- Strict-Transport-Security: max-age=31536000

You may need to safely load the site once

- Trust-on-first use

Browsers now also do HSTS-preloading
Other Mitigations

HTTP Public Key Pinning
https://en.wikipedia.org/wiki/HTTP_Public_Key_Pinning

Online Certificate Status Protocol
Apple Fail (https://gotofail.com/)

```c
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
    uint8_t *signature, UInt16 signatureLen)
{
    OSStatus err;
    SSLBuffer hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t hashes[SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN];
    SSLBuffer signedHashes;
    uint8_t *dataToSign;
    size_t dataToSignLen;

    ...  
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
    goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    err = sslRawVerify(ctx,
    ctx->peerPubKey,
    dataToSign,
    dataToSignLen,
    signature,
    signatureLen);

    if(err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify 
    "returned %d\n", (int)err);
        goto fail;
    }
}
```
CRIME Attack

Leverage compression to leak HTTP cookies
Need to be able to inject a script in a webpage
Issue multiple requests to target website to brute force cookie
Compression

Header sent with every request

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data

Slkgloirskjdal3irjndfsdndljsdpsidjsdp91jnfljdsf;9jas;ofdas;dqlnds

Original data → Compressed data → Encrypted data
Compression

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data
Cookie: sessionid=a

Saved transmission bandwidth due to compression
Compression

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data

Original data

Compressed data

Saved transmission bandwidth due to compression

Encrypted data

Observing the amount of data transmitted tells me when I get a match in the POST data.
Heartbleed
HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).

User Meg wants these 6 letters: POTATO. User 29 wants pages about '111 games'. Unlocking secure records with master key 51309573343.

POTATO

SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).

File ID 170 has a file '/bin/potato' in it. The user requested "33 files in car why". Note: Files For ID 375.381. 43.17 are in '/tmp/files-3843'. User Meg wants these 4 letters: BIRD. There are currently 34 connections open. User Brenden uploaded the file to the server.

HMM...
User Meg wants these 4 letters: BIRD. There are currently 335 connection types. User Brendan uploaded the file...