CSE509 System Security

SSL and TLS

Nick Nikiforakis
nick@cs.stonybrook.edu
Low-level
• The browser is a program written in memory-unsafe languages
• Has plugins written in memory-unsafe languages (Flash, Java, etc.)

Attacks
• Buffer overflows and overreads
  • Stack
  • Heap
• Integer overflows
• Execution of attacker-provided/desired code
  • Shellcode
  • Ret2libc
  • ROP

Web
• The browser is a container and execution environment of web applications

Attacks
• Against the web application
  • Client-side
  • Server-side
• Against the user
• Against the browser
  • Against web login
  • Against the underlying OS

Security basics (Principles / Ideas / Reoccurring mistakes)
What Is SSL / TLS?

• Secure Sockets Layer and Transport Layer Security protocols
  – Same protocol design, different crypto algorithms

• De facto standard for Internet security
  – “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”

• Deployed in every Web browser; also VoIP, payment systems, distributed systems, etc.
SSL / TLS Guarantees

• End-to-end secure communications in the presence of a network attacker
  – Attacker completely Owns the network: controls Wi-Fi, DNS, routers, his own websites, can listen to any packet, modify packets in transit, inject his own packets into the network

• Scenario: you are reading your email from an Internet café connected via a r00ted Wi-Fi access point to a dodgy ISP in a hostile authoritarian country
SSL and TCP

• SSL assumes that it lives on top of a transmission layer that will take care of resending lost packets and reordering packets
  – It doesn’t do the work of TCP. It relies on it being there

• SSL goals:
  – Detect alterations done by a MitM
  – Ensure data confidentiality
History of the Protocol

- SSL 1.0 – internal Netscape design, early 1994?
  - Lost in the mists of time
- SSL 2.0 – Netscape, Nov 1994
  - Several weaknesses
- SSL 3.0 – Netscape and Paul Kocher, Nov 1996
- TLS 1.0 – Internet standard, Jan 1999
  - Based on SSL 3.0, but not interoperable (uses different cryptographic algorithms)
- TLS 1.1 – Apr 2006
- TLS 1.2 – Aug 2008
SSL Basics

• SSL consists of two protocols

• Handshake protocol
  – Uses public-key cryptography to establish several shared secret keys between the client and the server

• Record protocol
  – Uses the secret keys established in the handshake protocol to protect confidentiality, integrity, and authenticity of data exchange between the client and the server
SSL Handshake Protocol

• Runs between a client and a server
  – For example, client = Web browser, server = website

• Negotiate version of the protocol and the set of cryptographic algorithms to be used
  – Interoperability between different implementations

• Authenticate server and client (optional)
  – Use digital certificates to learn each other’s public keys and verify each other’s identity
  – Often only the server is authenticated

• Use public keys to establish a shared secret
Handshake Protocol Structure

ClientHello

ServerHello, [Certificate], [ServerKeyExchange], [CertificateRequest], ServerHelloDone

[Certificate], ClientKeyExchange, [CertificateVerify]

switch to negotiated cipher

Finished

Record of all sent and received handshake messages

switch to negotiated cipher

Finished
ClientHello

Client announces (in plaintext):
Protocol version he is running
Cryptographic algorithms he supports
Fresh, random number
ClientHello (RFC)

struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
ServerHello

C, $\text{version}_c$, $\text{suites}_c$, $\text{N}_c$

ServerHello

Server responds (in plaintext) with:
Highest protocol version supported by both the client and the server
Strongest cryptographic suite selected from those offered by the client
Fresh, random number
ServerKeyExchange

Server sends his public-key certificate containing either his RSA, or his Diffie-Hellman public key (depending on chosen crypto suite)
The client generates secret key material and sends it to the server encrypted with the server’s public key (if using RSA).
struct {
    select (KeyExchangeAlgorithm) {
        case rsa: EncryptedPreMasterSecret;
        case diffie_hellman: ClientDiffieHellmanPublic;
    } exchange_keys
} ClientKeyExchange

struct {
    ProtocolVersion client_version;
    opaque random[46];
} PreMasterSecret

Where do random bits come from?

Random bits from which symmetric keys will be derived (by hashing them with nonces)
Debian Linux (2006-08)

- A line of code commented out from md_rand
  - \texttt{MD\_Update(&m,buf,j); /* purify complains */}
- Without this line, the seed for the pseudo-random generator is derived only from process ID
  - Default maximum on Linux = 32768
- Result: \textbf{all} keys generated using Debian-based OpenSSL package in 2006-08 are predictable
  - “Affected keys include SSH keys, OpenVPN keys, DNSSEC keys, and key material for use in X.509 certificates and session keys used in SSL/TLS connections”
“Core” SSL 3.0 Handshake

C, version\(_c\)=3.0, suites\(_c\), \(N_c\)

version\(_s\)=3.0, suitess\(_s\), \(N_s\), certificate for PK\(_s\), “ServerHelloDone”

\(\{\text{Secret}_c\}_{PK_s}\) if using RSA

C and S share secret key material (secret\(_c\)) at this point

switch to keys derived from secret\(_c\), \(N_c\), \(N_s\)

Finished

Finished
Version Rollback Attack

C, version\textsubscript{C} = 2.0, suites\textsubscript{C}, N\textsubscript{C}

Server is fooled into thinking he is communicating with a client who supports only SSL 2.0

version\textsubscript{S} = 2.0, suite\textsubscript{S}, N\textsubscript{S}, certificate for PK\textsubscript{S}, “ServerHelloDone”

\{Secret\textsubscript{C}\}_{PK\textsubscript{S}}

C and S end up communicating using SSL 2.0 (weaker earlier version of the protocol that does not include “Finished” messages)
SSL 2.0 Weaknesses (Fixed in 3.0)

• Cipher suite preferences are not authenticated
  – “Cipher suite rollback” attack is possible
• Weak MAC construction, MAC hash uses only 40 bits in export mode
• SSL 2.0 uses padding when computing MAC in block cipher modes, but padding length field is not authenticated
  – Attacker can delete bytes from the end of messages
• No support for certificate chains or non-RSA algorithms
“Chosen-Protocol” Attacks

• Why do people release new versions of security protocols? Because the old version got broken!
• New version must be backward-compatible
  – Not everybody upgrades right away
• Attacker can fool someone into using the old, broken version and exploit known vulnerabilities
  – Similar: fool victim into using weak crypto algorithms
• Defense is hard: must authenticate version early
• Many protocols had “version rollback” attacks
  – SSL, SSH, GSM (cell phones)
Version Check in SSL 3.0

C, version\(_c\) = 3.0, suites\(_c\), N\(_c\) → version\(_s\) = 3.0, suite\(_s\), N\(_s\), certificate for PK\(_s\), “ServerHelloDone”

C and S share secret key material secret\(_c\) at this point

“Embed” version number into secret

\(\{\text{version}\_c, \text{secret}\_c\}\)\(_{PK_s}\)

Check that received version is equal to the version in ClientHello

switch to key derived from secret\(_c\), N\(_c\), N\(_s\)

switch to key derived from secret\(_c\), N\(_c\), N\(_s\)
SSL/TLS Record Protection

Use symmetric keys established in the handshake protocol usually skipped because of recent attacks.

Application Data

Fragment

Compress

Add MAC

Encrypt

Append SSL Record Header
Most Common Use of SSL/TLS
HTTPS and Its Adversary Model

• HTTPS: end-to-end secure protocol for Web
• Designed to be secure against network attackers, including man-in-the-middle (MITM) attacks

• HTTPS provides encryption, authentication (usually for server only), and integrity checking
The Lock Icon

• Goal: identify secure connection
  – SSL/TLS is being used between client and server to protect against active network attacker

• Lock icon should only be shown when the page is secure against network attacker
  – Semantics subtle and not widely understood by users
  – Problem in user interface design
HTTPS Security Guarantees

• The origin of the page is what it says in the address bar
  – User must interpret what he sees – what about paypal-members.com?

• Contents of the page have not been viewed or modified by a network attacker
Combining HTTPS and HTTP

• Page served over HTTPS but contains HTTP
  – IE 7: no lock, “mixed content” warning
  – Firefox: New versions block active mixed content but allow passive mixed content
  – Safari: Old versions did not detect mixed content
  – Flash does not trigger warning in IE7 and old versions of FF

• Network attacker can now inject scripts, hijack session
Mixed Content: UI Challenges
Mixed Content and Network Attacks

• Banks: after login, all content served over HTTPS

• Developer error: somewhere on bank site write
  
  <script src=http://www.site.com/script.js> </script>
  
  – Active network attacker can now hijack any session (how?)

• Better way to include content:
  
  <script src=/www.site.com/script.js> </script>
  
  – Served over the same protocol as embedding page
HTTP → HTTPS and Back

• Typical (bad) pattern: HTTPS upgrade
  – Come to site over HTTP, redirect to HTTPS for login
  – Browse site over HTTP, redirect to HTTPS for checkout
• sslstrip: network attacker downgrades connection
  – Rewrite `<a href=https://...>` to `<a href=http://...>`
  – Redirect Location: https://... to Location: http://...
  – Rewrite `<form action=https://...>` to `<form action=http://...>`

Can the server detect this attack?
Will You Notice?

[Moxie Marlinspike]

Note: While this was true for many years, browsers are now changing their UIs with regard to HTTP and HTTPS. More in a few slides
Whose public key is used to establish the secure session?
Authenticity of Public Keys

Problem: How does Alice know that the public key she received is really Bob’s public key?
Distribution of Public Keys

• Public announcement or public directory
  – Risks: forgery and tampering

• Public-key certificate
  – Signed statement specifying the key and identity
    • \( \text{sig}_{\text{Alice}}(\text{“Bob”}, \text{PK}_B) \)

• Common approach: certificate authority (CA)
  – An agency responsible for certifying public keys
  – Browsers are pre-configured with 100+ of trusted CAs
  – A public key for any website in the world will be accepted by the browser if certified by one of these CAs
Trusted Certificate Authorities

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDC</td>
<td></td>
</tr>
<tr>
<td>TDC Internet</td>
<td></td>
</tr>
<tr>
<td>Thawte</td>
<td></td>
</tr>
<tr>
<td>Thawte Consulting</td>
<td></td>
</tr>
<tr>
<td>Thawte Consulting cc</td>
<td></td>
</tr>
<tr>
<td>thawte, Inc.</td>
<td></td>
</tr>
<tr>
<td>The Go Daddy Group, Inc.</td>
<td></td>
</tr>
<tr>
<td>The USERTRUST Network</td>
<td></td>
</tr>
<tr>
<td>TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş. ...</td>
<td></td>
</tr>
<tr>
<td>Unizeto Sp. z o.o.</td>
<td></td>
</tr>
<tr>
<td>ValiCert, Inc.</td>
<td></td>
</tr>
<tr>
<td>VeriSign, Inc.</td>
<td></td>
</tr>
<tr>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td>Wells Fargo</td>
<td></td>
</tr>
<tr>
<td>Wells Fargo WellsSecure</td>
<td></td>
</tr>
<tr>
<td>XRamp Security Services Inc</td>
<td></td>
</tr>
</tbody>
</table>
CA Hierarchy

• Browsers, operating systems, etc. have trusted root certificate authorities
  – Firefox 3 includes certificates of 135 trusted root CAs

• A Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
  – Certificate “chain of trust”
    • \( \text{sig}_{\text{Verisign}}(\text{“SBU”}, \text{PK}_{\text{SB}}), \text{sig}_{\text{SB}}(\text{“Nick N.”}, \text{PK}_{\text{Nick}}) \)

• CA is responsible for verifying the identities of certificate requestors, domain ownership
Certificate Hierarchy

What power do they have?

Who trusts their certificates?
Example of a Certificate

Important fields

Certificate Viewer: "mail.google.com"

General

Certificate Hierarchy
- GlobalSign Root CA - R2
  - Google Internet Authority G3
- mail.google.com

Certificate Fields
- Subject Public Key Algorithm
  - Algorithm Identifier
  - Algorithm Parameters
- Subject's Public Key
- Extensions
  - Extended Key Usage
  - Certificate Key Usage

Field Value
- Key size: 256 bits
- Base point order length: 256 bits
- Public value:
  04 db b9 97 03 69 04 2b 12 5b 60 fe 88 4f 67 49
  04 be 3a cd 9d 9d 9d e4 a9 85 8d 6f 00 b1 3b
  04 ef 9b 0f 87 5d 2b 3b a5 95 95 95 95 95 95 95
  04 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b
- Public value:
  04 db b9 97 03 69 04 2b 12 5b 60 fe 88 4f 67 49
  04 be 3a cd 9d 9d 9d e4 a9 85 8d 6f 00 b1 3b
  04 ef 9b 0f 87 5d 2b 3b a5 95 95 95 95 95 95 95
  04 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b 3b

Certificate Viewer: "mail.google.com"

General

Issued To
- Common Name (CN) mail.google.com
- Organization (O) Google LLC
- Organizational Unit (OU) <Not Part Of Certificate>
- Serial Number 14:1C:1B:A8:21:71:78:26

Issued By
- Common Name (CN) Google Internet Authority G3
- Organization (O) Google Trust Services
- Organizational Unit (OU) <Not Part Of Certificate>

Period of Validity
- Begins On Tuesday, October 23, 2018
- Expires On Tuesday, January 15, 2019

Fingerprints

Export...
Root Stores

• The place where all the certificates of trusted certificate authorities (root CA certificates) are held is called a “root store”

• Different operating systems have slightly different processes for approving new CAs but the processes typically boil down to:
  – Comply with various baselines for CAs
  – Comply with local laws
  – Pass yearly audits
  – Show business value to user base of the OS

• Most applications use the root stores provided by the underlying operating system
  – On MS Windows, you can access it by typing “certmgr.msc” in the “Run” dialogue

• Firefox is a notable exception since it ships with its own root store
  – Linux distributions commonly use this root store
International Domain Names

• Rendered using international character set
• Chinese character set contains characters that look like / ? = .
  – What could go wrong?
• Can buy a certificate for *.foo.cn, create any number of domain names that look like
  www.bank.com/accounts/login.php?q=me.foo.cn
  – What does the user see?
  – *.foo.cn certificate works for all of them!
Meaning of Color

What is the difference?

Domain Validation (DV) certificate vs. Extended Validation (EV) certificate

Means what?
Mobile Browsing

Google
www.google.com/

Gmail
mail.google.com/mail/#!

Bank of America Corporation
www.bankofamerica.com/privacy/


Same lock for DV and EV

Windows Phone 7: same behavior
... but only when URL bar present
... landscape mode: no URL bar
Extended Validation (EV) Certificates

• Certificate request must be approved by a human lawyer at the certificate authority
Questions about EV Certificates

• What does EV certificate mean?
• What is the difference between an HTTPS connection that uses a regular certificate and an HTTPS connection that uses an EV certificate?
• If an attacker has somehow obtained a non-EV certificate for bank.com, can he inject a script into https://bank.com content?
  – What is the origin of the script? Can it access or modify content that arrived from actual bank.com via HTTPS?
• What would the browser show – blue or green?
When Should The Lock Be Shown?

- All elements on the page fetched using HTTPS

For all elements:
- HTTPS certificate is issued by a certificate authority (CA) trusted by the browser
- HTTPS certificate is valid — means what?
- Common Name in the certificate matches domain name in the URL
X.509 Authentication Service

• Internet standard (1988-2000)
• Specifies certificate format
  – X.509 certificates are used in IPsec and SSL/TLS
• Specifies certificate directory service
  – For retrieving other users’ CA-certified public keys
• Specifies a set of authentication protocols
  – For proving identity using public-key signatures
• Can use with any digital signature scheme and hash function, but must hash before signing

Remember MD5?
Back in 2008

- Many CAs still used MD5
  - RapidSSL, FreeSSL, TrustCenter, RSA Data Security, Thawte, verisign.co.jp
- Sotirov et al. collected 30,000 website certificates
- 9,000 of them were signed using MD5 hash
- 97% of those were issued by RapidSSL

[Sotirov et al. “Rogue Certificates”]
Colliding Certificates

[Sotirov et al. “Rogue Certificates”]

<table>
<thead>
<tr>
<th>set by the CA</th>
<th></th>
<th>chosen prefix (difference)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>serial number</td>
<td></td>
<td></td>
<td>serial number</td>
<td></td>
</tr>
<tr>
<td>validity period</td>
<td></td>
<td></td>
<td>validity period</td>
<td></td>
</tr>
<tr>
<td>real cert domain name</td>
<td></td>
<td></td>
<td>rogue cert domain name</td>
<td></td>
</tr>
</tbody>
</table>

Hash to the same MD5 value!

Valid for both certificates!

- Real cert RSA key
- X.509 extensions
- Signature

- Collision bits (computed)
- Identical bytes (copied from real cert)
- Signature

X.509 extensions
Generating Collisions

[Sotirov et al. “Rogue Certificates”]

1-2 days on a cluster of 200 PlayStation 3’s

Equivalent to 8000 desktop CPU cores or $20,000 on Amazon EC2
Creating a Fake Intermediate CA

[Sotirov et al. “Rogue Certificates”]

<table>
<thead>
<tr>
<th>serial number</th>
<th>rogue CA cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>validity period</td>
<td>rogue CA RSA key</td>
</tr>
<tr>
<td>real cert domain name</td>
<td>rogue CA X.509 extensions</td>
</tr>
<tr>
<td>real cert RSA key</td>
<td>Netscape Comment Extension (contents ignored by browsers)</td>
</tr>
<tr>
<td>X.509 extensions</td>
<td>signature</td>
</tr>
<tr>
<td>signature</td>
<td>signature</td>
</tr>
</tbody>
</table>

chosen prefix (difference)

collision bits (computed)

identical bytes (copied from real cert)

We are now an intermediate CA. W00T!

CA bit!
Result: Perfect Man-in-the-Middle

[Sotirov et al. “Rogue Certificates”]

• This is a “skeleton key” certificate: it can issue fully trusted certificates for any site (why?)

• To take advantage, need a network attack
  – Insecure wireless, DNS poisoning, proxy auto-discovery, hacked routers, etc.
A Rogue Certificate
“Flame” malware

• Cyber-espionage virus (2010-2012)
• Signed with a fake intermediate CA certificate that appears to be issued by Microsoft and thus accepted by any Windows Update service
  – Fake intermediate CA certificate was created using an MD5 chosen-prefix collision against an obscure Microsoft Terminal Server Licensing Service certificate that was enabled for code signing and still used MD5
• MD5 collision technique possibly pre-dates Sotirov et al.’s work
  – Evidence of state-level cryptanalysis?
Hello

Here is my certificate

Validate the certificate
SSL/TLS Handshake

Hello

I am Chase.com
Here is my certificate

Issued by GoDaddy to
AllYourSSLAreBelongTo.us

Ok!
“Researchers at the University of Texas at Austin and Stanford University have discovered that poorly designed APIs used in SSL implementations are to blame for vulnerabilities in many critical non-browser software packages. Serious security vulnerabilities were found in programs such as Amazon’s EC2 Java library, Amazon’s and PayPal’s merchant SDKs, Trillian and AIM instant messaging software, popular integrated shopping cart software packages, Chase mobile banking software, and several Android applications and libraries. SSL connections from these programs and many others are vulnerable to a man in the middle attack...”

- Threatpost (Oct 2012)
Here is PayPal’s certificate
And here is my signed Diffie-Hellman value

... verify the signature on the DH value using
the public key from the certificate

```c
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(...);

Signature is verified here
```

Complete Fail Against MITM

- Discovered in February 2014
- All OS X and iOS software vulnerable to man-in-the-middle attacks
  - Broken TLS implementation provides no protection against the very attack it was supposed to prevent
- What does this tell you about quality control for security-critical software?
Certificate Revocation

• Revocation is very important
• Many valid reasons to revoke a certificate
  – Private key corresponding to the certified public key has been compromised
  – User stopped paying his certification fee to the CA and the CA no longer wishes to certify him
  – CA’s certificate has been compromised!
• Expiration is a form of revocation, too
  – Many deployed systems don’t bother with revocation
  – Re-issuance of certificates is a big revenue source for certificate authorities
Certificate Revocation Mechanisms

• Online revocation service
  – When a certificate is presented, recipient goes to a special online service to verify whether it is still valid

• Certificate revocation list (CRL)
  – CA periodically issues a signed list of revoked certificates
  – Can issue a “delta CRL” containing only updates

Q: Does revocation protect against forged certificates?
X.509 Certificate Revocation List

Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate.
Some Questions About Certificates

• How do CAs verify identities of domains to whom they issue certificates (domain validation)?
• Does your browser check whether the site’s certificate has been revoked?
• What do you do when your browser warns you that the site’s certificate has expired?
  – Most users click through, enter credentials
• Over 40% of certs are self-signed – means what?
Invalid Certificate Warnings

Some CA horror stories...
Comodo

• Comodo is one of the trusted root CAs
  – Its certificates for any website in the world are accepted by every browser

• Comodo accepts certificate orders submitted through resellers
  – Reseller uses a program to authenticate to Comodo and submit an order with a domain name and public key, Comodo automatically issues a certificate for this site
Comodo Break-In

• An Iranian hacker broke into instantSSL.it and globalTrust.it resellers, decompiled their certificate issuance program, learned the credentials of their reseller account and how to use Comodo API
  – username: gtadmin, password: globaltrust
• Wrote his own program for submitting orders and obtaining Comodo certificates
• On March 15, 2011, got Comodo to issue 9 rogue certificates for popular sites
  – Including: mail.google.com, login.live.com, login.yahoo.com, login.skype.com, addons.mozilla.org
Consequences

• Attacker needs to first divert users to an attacker-controlled site instead of Google, Yahoo, Skype, but then...
  – For example, use DNS to poison the mapping of mail.yahoo.com to an IP address
• ... “authenticate” as the real site
• ... decrypt all data sent by users
  – Email, phone conversations, Web browsing

Q: Does HTTPS help? How about EV certificates?
DigiNotar Break-In

- In June 2011, the same “ComodoHacker” broke into a Dutch certificate authority, DigiNotar
  - Message found in scripts used to generate fake certificates:
    “THERE IS NO ANY HARDWARE OR SOFTWARE IN THIS WORLD EXISTS WHICH COULD STOP MY HEAVY ATTACKS MY BRAIN OR MY SKILLS OR MY WILL OR MY EXPERTISE"

- Security of DigiNotar servers
  - All core certificate servers in a single Windows domain, controlled by a single admin password (Pr0d@dm1n)
  - Software on public-facing servers out of date, unpatched
  - Tools used in the attack would have been easily detected by an antivirus... if it had been present
Consequences of DigiNotar Hack

• Break-in not detected for a month
• Rogue certificates issued for *.google.com, Skype, Facebook, www.cia.gov, and 527 other domains
• 99% of revocation lookups for these certificates originated from Iran
  – Evidence that rogue certificates were being used, most likely by Iranian government or Iranian ISPs to intercept encrypted communications
    • Textbook man-in-the-middle attack
  – 300,000 users were served rogue certificates
• DigiNotar filed for bankruptcy
• In Feb 2012, admitted issuance of an intermediate CA certificate to a corporate customer
  – Purpose: “re-sign” certificates for “data loss prevention”
  – Translation: forge certificates of third-party sites in order to spy on employees’ encrypted communications with the outside world
• Customer can now forge certificates for any site in world... and they will be accepted by any browser!
  – What if a “re-signed” certificate leaks out?
• Do other CAs do this?
TurkTrust

• In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
  – TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
  – Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to filter SSL traffic from its network
• This rogue *.google.com certificate was trusted by every browser in the world
What can be done against rogue Certificate Authorities?
Certificate Transparency

• Experimental open standard for monitoring and auditing digital certificates
  – This was put in place after many incidents where certificate authorities were found to be issuing certificates that they were not supposed to

• A CA that participates in Certificate Transparency adds to a global append-only log an entry for each new certificate that it issues
  – The idea is that others can monitor this log and sound an alarm when domain certificates were created without the knowledge of the appropriate domain owners

• Google Chrome requires Certificate Transparency for all EV (Extended validation) certificates
  – No CT, no green address bar

• Eventually Google Chrome will expect this from all CAs
Economics of SSL

• Up until a few years ago, SSL/TLS was something that only websites handling sensitive data would need to get
  – Banks
  – Social Networks
  – Email providers
  – Etc.

• Even though the price of certificates has gone down, today one would need to pay between $10 and $100 for a simple domain-verified (DV) certificate
  – Wildcard certificates (e.g. *.example.com) cost more
  – Symantec, GeoTrust, RapidSSL, Thawte are all common providers
Economics of SSL

• Process for a traditional DV certificate
  1. Go to a commercial certificate provider
  2. Pay the fee
  3. Prove that you own the domain
     • Receive email on an address listed in your WHOIS information
     • Place a file on your root directory with a specific file-name and file-content
     • Add a DNS record specified by the SSL provider
  4. Generate a Certificate Signing Request on your system (e.g. using openssl) and a private key and upload the CSR to the certificate provider
  5. Receive your certificate in an email
  6. Set up your webserver to use that certificate
Economics of SSL

• This was the only way of doing things up until 2016 when Let’s Encrypt became publicly available

• Let’s Encrypt is a certificate authority that provides free domain-verified certificates
  – They were not the first cost-free provider but they had the backing and PR of large companies
  – They provided software that makes the fetching and installation of a certificate, essentially automatic
Let’s Encrypt

- Let’s Encrypt uses a challenge-response protocol called ACME (Automated Certificate Management Environment) to verify that a client is indeed the owner of a domain name, before it issues a free certificate.

- The most popular implementation of ACME is that of certbot
  - [https://certbot.eff.org/](https://certbot.eff.org/)
Let’s Encrypt growth
Everyone uses Let’s Encrypt...including the bad guys

• Since everyone can now have an SSL certificate, so can the bad guys
  – https://www.totallypaypal.com

• Websites with SSL have been traditionally thought of as “secure” and “safe” so phishing attacks may work better on SSL websites
Let’s Encrypt’s response

• Certificate Authorities are not supposed to be content watchdogs
  – A CA should issue a DV certificate to anyone who can prove ownership of a given domain
    • https://letsencrypt.org/2015/10/29/phishing-and-malware.html

• This will require re-educating users about the meaning of https
  – “Secure” and “safe” are not the same thing
  – HTTPS guarantees “secure” but does not say anything about “safe”
To summarize the differences between the two

HTTPS & SSL doesn't mean "trust this." It means "this is private." You may be having a private conversation with Satan.

9:08 AM - 4 Apr 2012

3,811 Retweets 3,100 Likes
Vision for the web and Let’s Encrypt

• By removing the financial cost and technical-know-how barriers, the backers of Let’s Encrypt are pushing for an HTTPS-only web
  – This is the opposite of what we used to have a few years ago

• Google and Firefox have already started penalizing websites that do not use SSL/TLS and these penalties will become even more severe in the coming years
Google Chrome

Treatment of HTTP pages with password or credit card form fields:

- **Current (Chrome 53)**: 
  - [i] login.example.com

- **Jan. 2017 (Chrome 56)**:
  - [i] Not secure | login.example.com

Treatment of HTTP pages outside Incognito mode:

- **Current (Chrome 58)**: 
  - [i] example.com

- **Oct. 2017 (Chrome 62) at page load**:
  - [i] example.com

- **Oct. 2017 (Chrome 62) when entering data**:
  - [i] Not secure | example.com

Treatment of HTTP pages in Chrome Incognito mode:

- [i] example.com
- [i] Not secure | example.com
- [i] Not secure | example.com
Firefox

Any regular web page

A page with password input fields
Not only UI penalties

• Google Chrome engineers are also pushing for making powerful features unavailable on websites that do not utilize HTTPS

• Current list of features
  – Geolocation (already not available in non-HTTPS websites)
  – Device motion/orientation
  – Notifications
  – AppCache
  – getUserMedia

• Google search has also been using SSL support as a ranking feature
  – If your site does not support HTTPS, all other things being equal, it will be lower-ranked than those websites that do support HTTPS
HTTPS interception

• There are cases (acceptable/less acceptable) where one would want to be able to inspect HTTPS traffic
  – Antivirus software wants to be able to inspect your traffic, even if it is encrypted
  – Companies want to inspect traffic to ensure that confidential data is not being leaked or that you are not accessing websites that you are not supposed to
    • Destination domain is available during SSL handshake but no paths or URL parameters
  – Malware/adware located on your machine wants to be able to inject content despite the presence of HTTPS
HTTPS interception

• Methods of intercepting HTTPS traffic
  – Use browser extensions. These operate on already decrypted traffic.
  • Downsides
    – Extension APIs are limited (on purpose)
    – Users can typically uninstall extensions from their browsers
  – Install a root certificate in the utilized browser
    • Possible for Firefox
  – Install a root certificate in trusted root store of the system
    • Can be done by the administrator in a company
    • Or the AV/malware that you just installed
HTTPS without interception

Image source: Dell Secure works
HTTPS with interception

The interception proxy can be:
- A piece of software on the client endpoint
- A network device that intercepts the client’s traffic before it reaches the outside world
HTTPS without interception

Understanding domain squatting
Measurement and countermeasures against unwanted tracking
Assessment of the security of remote servers using mostly passive techniques
Client-side defense mechanisms against phishing and web application attacks

I received my B.Sc. ('07) and M.Sc. ('09) in Computer Science from the University of Crete, and
HTTPS with interception by Fiddler

have been the most active in are:

- Analysis of online ecosystems
- Understanding domain squatting
- Measurement and countermeasures against unwanted tracking
- Assessment of the security of remote servers using mostly passive techniques
- Client-side defense mechanisms against phishing and web application attacks

Security overview

This page is secure (valid HTTPS).

Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by DO_NOT_TRUST_FiddlerRoot.
This can go terribly wrong

• In 2015, it was discovered that all Lenovo laptops sold between October and December 2014 were pre-installed with software from a company called Superfish
  – 750K machines sold in the US

• This software installed its own root certificate in the trust store of Windows allowing the software to inject ads on HTTPS-encrypted websites
  – This was obviously never told to the user
  – Textbook example of “adware”
Superfish intercepting bankofamerica.com connections
Universal keys

• Instead of generating a new certificate for each machine (like Fiddler does), Superfish had the same certificate on all machines.

• Researchers were able to reverse-engineer the adware and extract its private key which meant that they could do a silent MITM attack on all Lenovo users that had this installed.
  - Silent: No browser warnings, green lock, everything works as expected.

• Windows issued updates to remove the offending certificate and the FTC recently (September 2017) settled with Lenovo for $3.5 million and increased audits on software preinstalled on its machines.
Summary

• HTTP is by default plaintext
  – Everyone with access “to the wire”, can inspect and change HTTP requests and responses

• SSL/TLS is a suit of protocols for providing confidentiality and integrity of TCP content
  – Typical usage: Use asymmetric crypto to exchange a temporal symmetric-crypto key

• Differences in certificates (domain verification versus extended verification)

• Mistakes can and have happened
  – CA compromises
  – Over-eager device manufacturers

• Certificate Transparency can be used to detect rogue CAs

• Economics of SSL and Let’s Encrypt, push for an all-encrypted web

• HTTPS interception, what it is, why companies want it, why it can be bad
Questions?
Credits

• Original slide deck by Vitaly Shmatikov
• Modified and Extended by Nick Nikiforakis