Lecture 22 – CAs and HTTPS Attacks

Stephen Checkoway
University of Illinois at Illinois
CS 487– Fall 2017
Slides from Miller and Bailey's ECE 422

Certificates

•Make use of trusted **"Certificate Authorities" (CA)**

- "This public key with SHA-256 hash (XXX) belongs to the site (name, e.g., Amazon.com)"
  - **Digitally signed** by a certificate authority

- Your browsers (e.g., Firefox, Chrome) trust a specific set of CAs as root CAs
  - Shipped with the public keys of the root CAs
  - Why do we need more than 1?

How the CA verifies your identity

- Typically 'DV' (**domain** verification)
  - –Proves you are in control of DNS registration
  - –Just an email based challenge to the address in the domain registration records
    - •Or some default email address, admin@domain.com
    - •Minimally secure [Why?]
  - –Alternately a web-based challenge
    - –Include challenge response in a <meta> tag
- Cert has an expiration date (e.g., one year ahead) [Why?]

How to invalidate certificates?

- Expiration date of certs
- Certificate revocation
- What happens if a CA's secret key is leaked?
  - Can we trust the old certs from that CA?
- Interesting fact:
  - Google has instrumented Chrome such that when it observes a certificate for Google.com that it doesn't recognize, it panics…. (has happened several times)

Self-signed Certificates

- Issuer signs their own certificate
  - A loop in the owner and signer
- Avoid CA fees, useful for testing
  - You can add yourself as a CA to your own browser
- Browsers display warnings that users have to override
- Protects only against passive attacker “optimistic encryption”


Not secure
https://pcwebshop.co.uk


This Connection is Untrusted
You have asked Firefox to connect securely to [redacted], but we can't confirm that your connection is secure.
Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.
What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.
Get me out of here!
Technical Details
I Understand the Risks

TLS Certificates

- A trusted authority vouches that a certain public key belongs to a particular site
- Format called x.509 (complicated)
- Browsers ship with CA public keys for large number of trusted CAs [accreditation process]
- Important fields:
    - Common Name (CN) [e.g., *.google.com]
      Expiration Date [e.g. 2 years from now]
      Subject's Public Key
      Issuer -- e.g., Verisign
      Issuer's signature
- Common Name field
    - Explicit name, e.g. ece.illinois.edu
    - Or wildcard, e.g. *.illinois.edu

X509 Certificates

Subject: C=US/O=Google Inc/CN=www.google.com
Issuer: C=US/O=Google Inc/CN=Google Internet Authority
Serial Number: 01:b1:04:17:be:22:48:b4:8e:1e:8b:a0:73:c9:ac:83
Expiration Period: Jul 12 2010 - Jul 19 2012
Public Key Algorithm: rsaEncryption
Public Key: 43:1d:53:2e:09:ef:dc:50:54:0a:fb:9a:f0:fa:14:58:ad:a0:81:b0:3d
7c:be:b1:82:19:b9:7c3:8:04:e9:1e5d:b5:80:af:d4:a0:81:b0:b0:68:5b:a4:a4
:ff:b5:8a:3a:a2:29:e2:6c:7c3:8:04:e9:1e5d:b5:7c3:8:04:e9:39:23:46

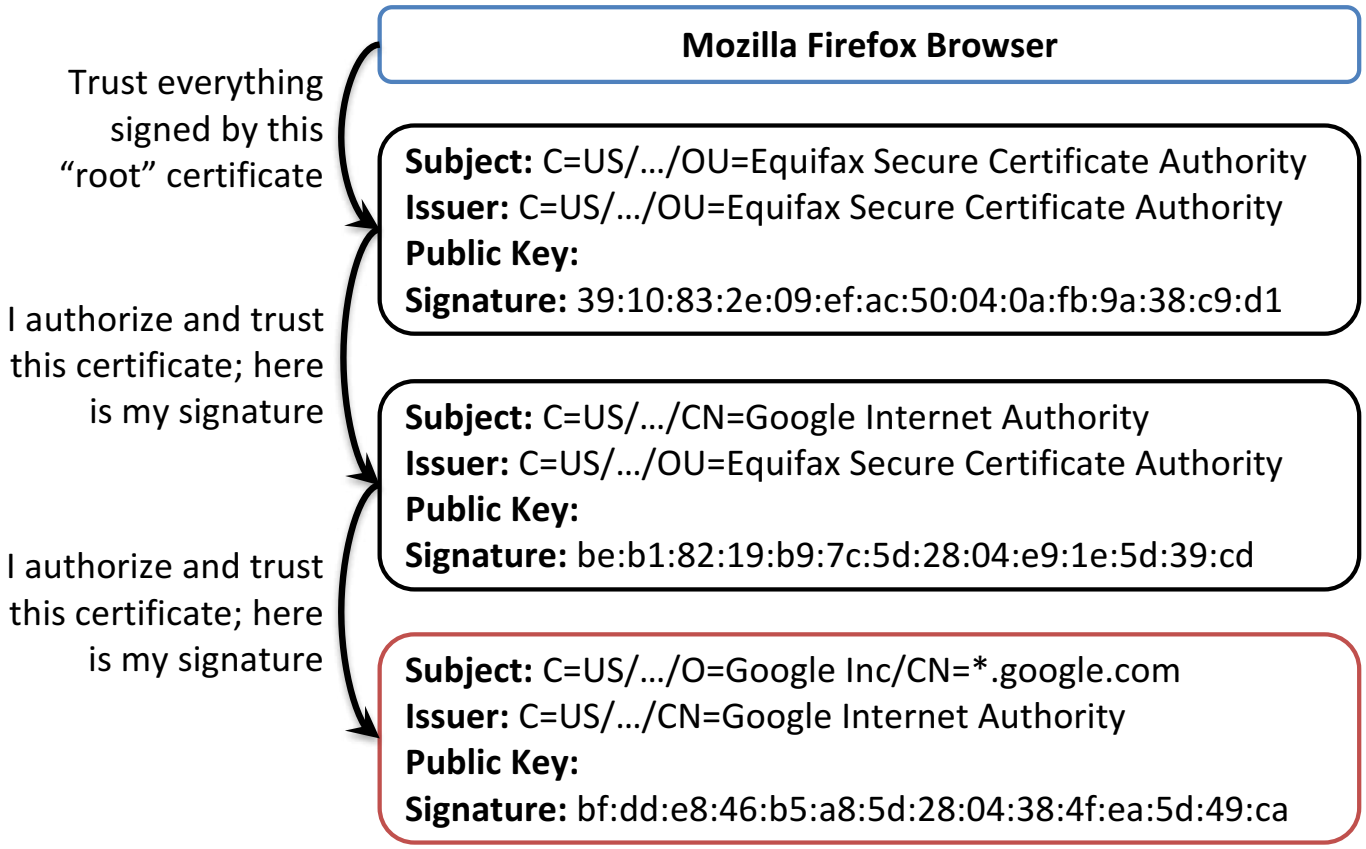
Signature Algorithm: sha1WithRSAEncryption

Signature: 39:10:83:2e:09:ef:ac:50:04:0a:fb:9a:f0:fa:14:58:ad:a0:81:b0:3d
7c:be:b1:82:19:b9:7c3:8:04:e9:1e5d:b5:80:af:d4:a0:81:b0:b0:68:5b:a4:a4
:ff:b5:8a:3a:a2:29:e2:6c:7c3:8:04:e9:1e5d:b5:7c3:8:04:e9:1e:5d:b5

Certificate Chains

- CA can delegate ability to generate certificates for certain names: Intermediate CAs
- Root CA signs "certificate issuing certificate" for delegated authority
- Browser that trusts root can examine certs to establish validity -- "Chain of trust”
- How to find out about all the CAs?
- More than 1000 trusted parties today, can sign for any domain – huge problem!

Certificate Chains



Certificate Authority Ecosystem

Each browser trusts a set of CAs

- CAs can sign certificates for new CAs
- CAs can sign certificates for any web site

If a single CA is compromised, then the entire system is compromised

We ultimately place our complete trust of the Internet in the weakest CA

Immediate Concerns

- Nobody has any idea who all these CAs are…
- 1,733 *umich*-known browser trusted CAs
- History of CAs being hacked (e.g. Diginotar)
- Oooops, Korea gave every elementary school, library, and agency a CA certificate (1,324)
  - Luckily invalid due to a higher-up constraint

Getting a Certificate

- Certificates are free (from LetsEncrypt!)
  - Identity validated by challenge to website
- Certificates are cheap elsewhere too
  - Identity is validated via e-mail to the default e-mail addresses
- Setting up SSL is hard. People are terrible at it.
  - Certificate Signing Requests, eugh
  - Integrating in a web server

TLS in the browser

- Lock icon
  - HTTPS cert must be issued by a CA trusted by browser (or chain to one that is)
  - HTTPS cert is valid (e.g., not expired or revoked)
  - CommonName in cert matches domain in URL
- Extended Validation (EV) certificates
  - CA does extra work to verify identity -- expensive, but NO more secure
- Invalid certificate warnings

Attack Vectors

- Attack the weakest Certificate Authority
- Attack browser implementations
- Magically notice a bug in a key generation library that leads you to discovering all the private keys on the Internet
- Attack the cryptographic primitives
  - Math is hard

Search

About 274,000 results (0.24 seconds)

Everything
Images
Maps
Videos
News
Shopping
More

All results
Related searches
More search tools

-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...
pastebin.com/TbaeU93m
19 Apr 2010 – **...** the difference. Copied. -----**BEGIN RSA PRIVATE KEY**-----.
MIICXwIBAAKBpenis1ePqHkVN9IKaGBESjV6zBrIsZc+XQYTtSlVa9R/4SAXoYpl **...**

-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...
pastebin.com/sC7bGw30
18 Apr 2010 – **...** difference. Copied. -----**BEGIN RSA PRIVATE KEY**-----.
MIIEogIBAAKCAQEAvxBalhzKMewLvmIr1ptID1gO7EWGFyudzOAHLqm3+0+gpPbk **...**

site:pastebin.com "-----**BEGIN RSA PRIVATE KEY**-----" - Posterous
cdevers.posterous.com/sitepastebincom-**begin-rsa-private-key**-google
20 Apr 2010 – Apr 19, 2010 ... -----**BEGIN RSA PRIVATE KEY**-----
MIICXwIBAAKBpenis1ePqHkVN9IKaGBESjV6zBrIsZc+ XQYTtSlVa9R/4SAXoYpl .

help/en/howto/sftp – Cyberduck
trac.cyberduck.ch/wiki/help/en/howto/sftp
Private keys containing a DSA or RSA private key in PEM format are supported (look for -----BEGIN DSA PRIVATE KEY----- or -----**BEGIN RSA PRIVATE KEY**----- **...**

SSH access with a private RSA key [Archive] - VanDyke Software For…
forums.vandyke.com/archive/index.php/t-2185.html
2 Sep 2011 – -----**BEGIN RSA PRIVATE KEY**-----
MIIEogIBAAKCAQBujdbtxyIX4KaQPeTf5F/
aOSBwSpZN4MjTixU2Yq8JkipjMYpYwpNj1TODzRJf **...**

Attacking site design

- SSLstrip attack
  - Proxy through the content w/o HTTPS
- Defense
  - Default HTTPS for all web sites?
  - HSTS (hypertext strict transport security): header says: always expect HTTPS, enforced by browsers.
  - HTTPS Everywhere: browser extension
  - ~~EV: Extended Validation (compared to DV: Domain Validation)~~

Attacking site design

- Mixed Content attack -- Page loads over HTTPS but contains content over HTTP
  - e.g. JavaScript, Flash
  - Active attacker can tamper with HTTP content to hijack session
- Defense: Browser warnings: ["This page contains insecure content"],
  - but inconsistent and often ignored

UI interface based attacks

- Invalid certs
  - Expired, Common Name != URL, unknown CA (e.g., self-signed)
- Defense: browser warnings, anti-usability to bypass…
- Picture-in-picture attack: spoof the user interface
  - Attacker page draws fake browser window with lock icon
- Defense: individualized image

Attacking the PKI: CA compromise
Example: DigiNotar

Attacking the PKI: CA compromise
Example: DigiNotar

- DigiNotar ***was*** a Dutch Certificate Authority
- On June 10, 2011, **`*.google.com`** cert was issued to an attacker and subsequently used to orchestrate MITM attacks in Iran
- Nobody noticed the attack until someone found the certificate in the wild… and posted to *pastebin*

DigiNotar Contd.

- DigiNotar later admitted that dozens of fraudulent certificates were created
- Google, Microsoft, Apple and Mozilla all revoked the root Diginotar certificate
- Dutch Government took over Diginotar
- Diginotar went bankrupt and died

Attacking the PKI: Hash collisions

- MD5/SHA1 is known to be broken -- Can generate collisions
- In 2008, researchers showed that they could create a rogue CA certificate using an MD5 collision
- Attack: Make colliding messages A, B, with same MD5 hash:
  - A: Site certificate: "cn=attack.com, pubkey=....”
  - B: Delegated CA certificate: "pubkey=.... is allowed to sign certs for *”
  - Get CA to sign A -- Signature is Sign(MD5(message))
  - Signature also valid for B (same hash)
  - Attacker is now a CA!
  - Make a cert for any site, browsers will accept it

MD5 considered harmful

- MD5 CA certificates still exist, but CAs have stopped signing certificates with them
  - 879,705 certificates still have MD5 signatures

- SHA-1 should not be used either
  - 46,969,095 out of 146,442,087 certs ever seen by Censys use SHA1WithRSA (32%)

Attacking implementations: Null Termination Attack

- ASN.1 utilizes Pascal-style strings
- Web browsers utilize use C-style strings
- Announced by Moxie Marlinspike in 2009

`gmail.com\0.badguy.com`

Null Termination Attack

- www.attacker.com
  - [CAs verify cert by looking up who owns the last part of the domain via DNS record]
  - emails "webmaster@attack.com" --> "Click here to validate cert request"
- x.509 certs encode CN field as a Pascal string (length+data)
- Browsers copy it into a C string (data+\0)
- What if CA contains "\0"?
  - www.paypal.com\0.attacker.com?
  - CA contacts "attacker.com" to verify (last part of domain name)
  - Browsers copy to C string, terminates at "\0" -- see only paypal.com
  - Attacker now has a cert that works for Paypal!

Other implementation-based attacks

- Goto fail, Feb. 2014 (Apple SSL bug; skipped certificate check for almost a year!)
- Heartbleed, April 2014 (OpenSSL bug; leaked data, possibly including private key!)
- Mozilla BERserk vulnerability, Oct 2014 (Bug in verifying cert signatures, allowed spoofing certs, probably since the beginning….!)
- Logjam, Oct 2016 (TLS vulnerable to Man-in-the middle "Downgrade" attack)

Who controls the TLS endpoint?

Cloudbleed - (the other big news last week)
- one of the most popular “content delivery networks”
- acts as the SSL endpoint for many servers
- a **buffer overflow** attack caused it to leak HTTPS data

Clientside HTTP Interception -
- Most antivirus software intercepts your HTTPS [How?]
- Introduces new vulnerabilities by implementing poorly

Takeaways

- Use HTTPS! It’s so much better than nothing

- TLS keeps breaking. Use it, but don't rely on it exclusively.