



MANICODE
SECURE CODING EDUCATION

SSL / TLS / HTTPS Best Practices

JIM MANICO

Secure Coding Instructor

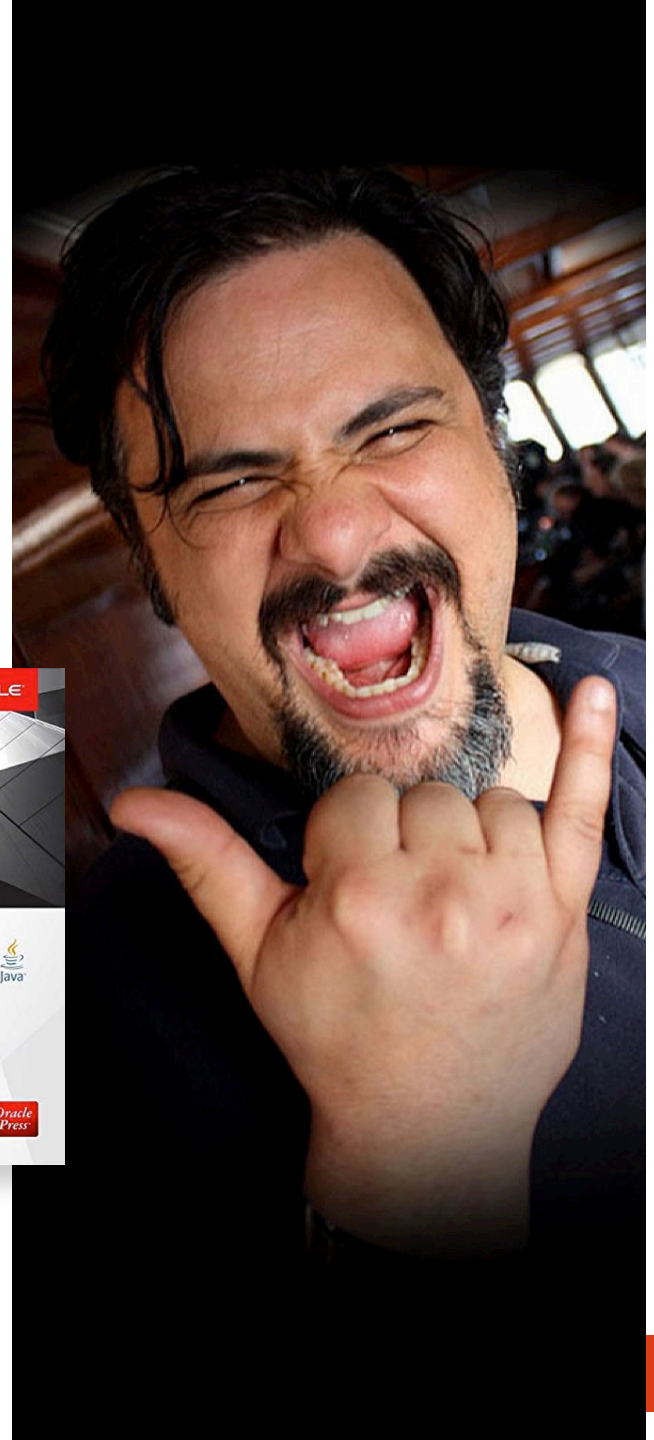
www.manicode.com

A little background dirt...

jim@manicode.com

 [@manicode](https://twitter.com/manicode)

- Former OWASP Global Board Member
- Project manager of the OWASP Cheat Sheet Series and several other OWASP projects
- 20+ years of software development experience
- Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press
- Kauai, Hawaii Resident





WARNING: Please do not attempt to hack any computer system without legal permission to do so. Unauthorized computer hacking is illegal and can be punishable by a range of penalties including loss of job, monetary fines and possible imprisonment.

ALSO: The *Free and Open Source Software* presented in these materials are examples of good secure development tools and techniques. You may have unknown legal, licensing or technical issues when making use of *Free and Open Source Software*. You should consult your company's policy on the use of *Free and Open Source Software* before making use of any software referenced in this material.

Where are we going?

What is SSL / TLS / HTTPS?

TLS Configuration

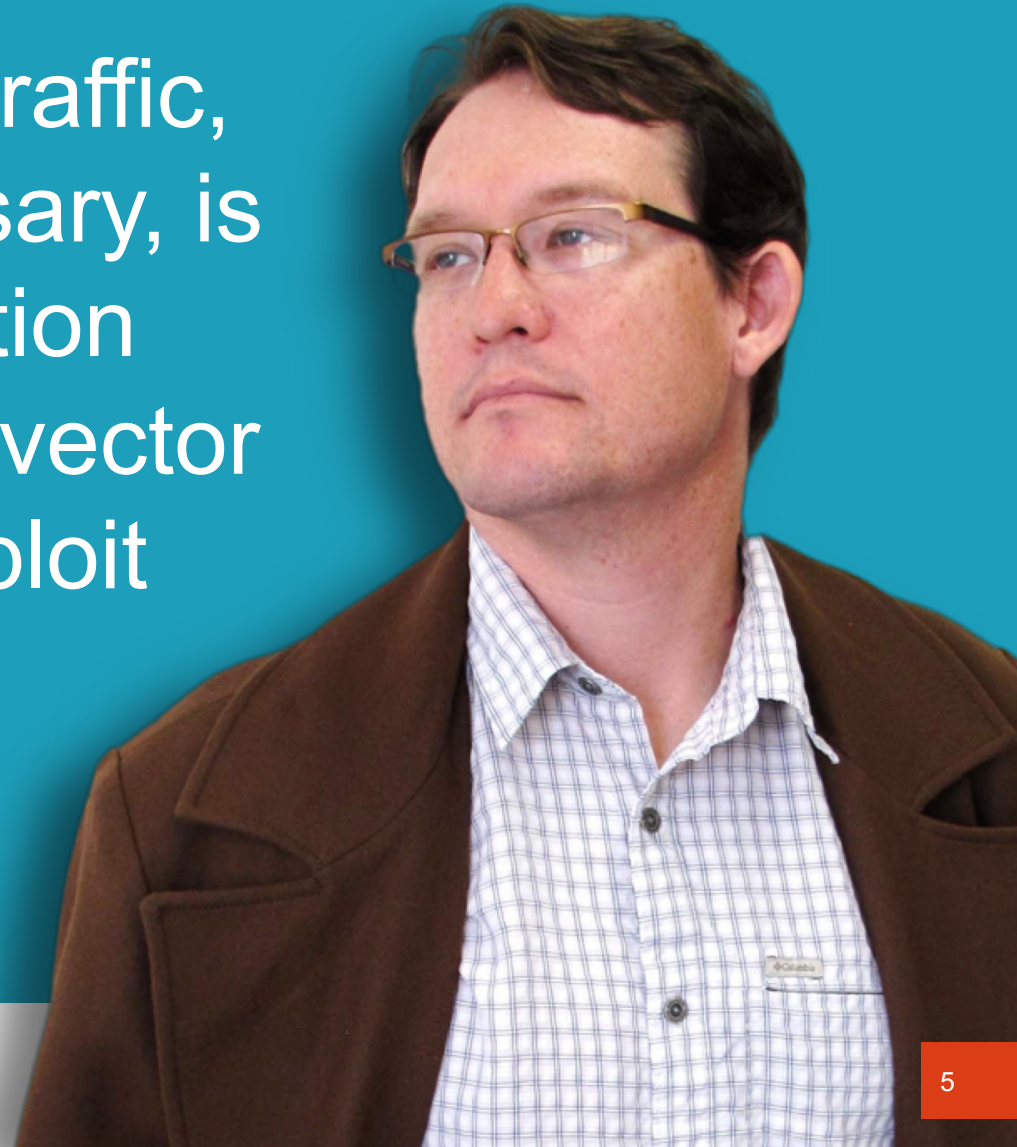
Strict Transport Security (HSTS)

Forward Secrecy

Certificate Pinning

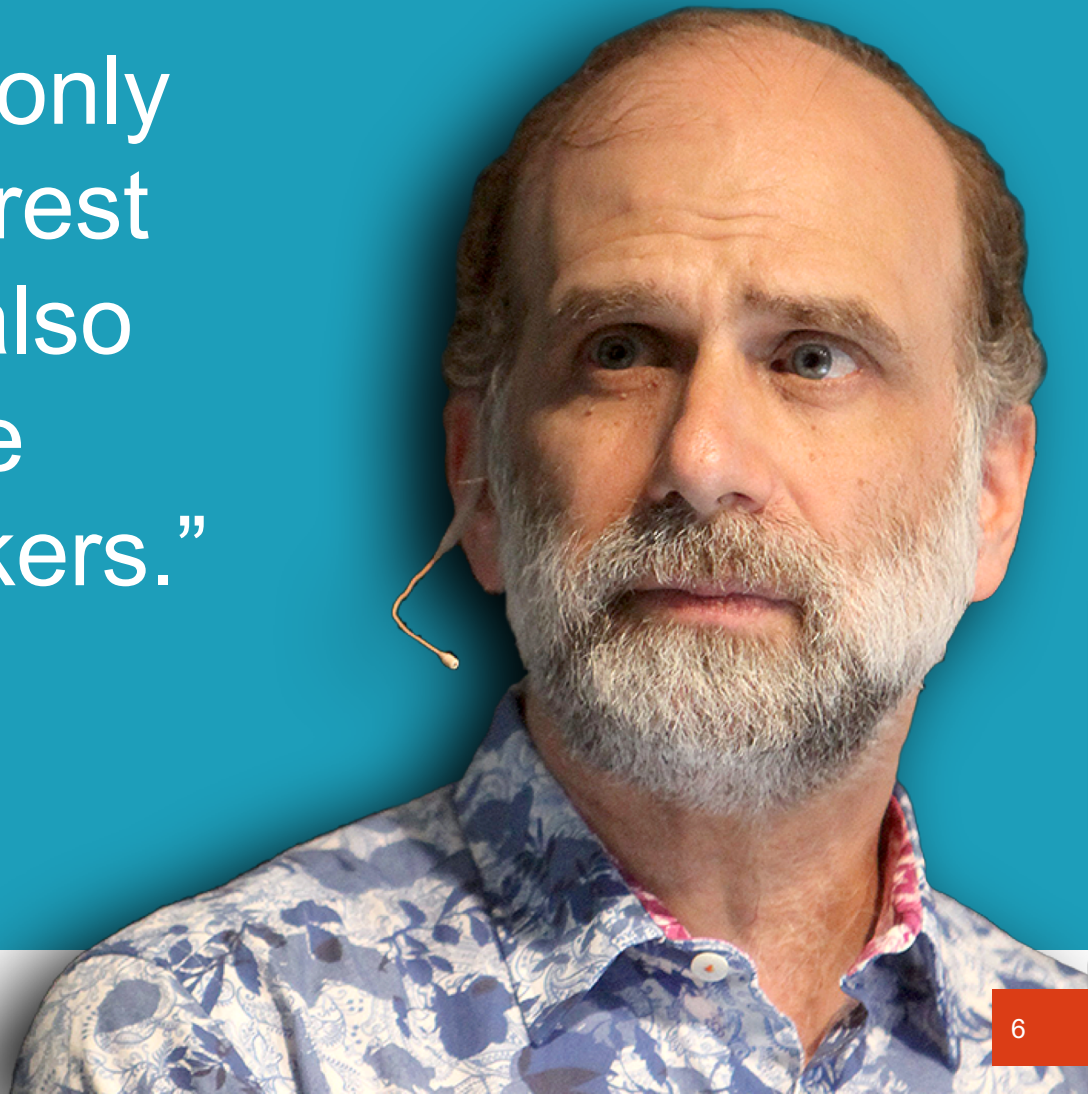
“Any unencrypted traffic, visible to an adversary, is not just an information leak, but an attack vector they can use to exploit your systems.”

Nick Weaver



“Cryptography is only truly useful if the rest of the system is also sufficiently secure against the attackers.”

Bruce Schneier
Security Engineering



Bypassing TLS

Cryptography is solid but the implementation has flaws

- Browser fail open policy
- Too much trust on certificate authorities
- Revocation capability is limited and doesn't scaled

Some of the most popular algorithms need to be decommissioned

Cryptography is **bypassed**,
not attacked

SSS

Transport Keys & Security

What's another way of looking at it?

Confidentiality

Spy cannot view your data

Integrity

Spy cannot change your data

Authenticity

Server you are visiting is the right one,
backed up by the Certificate Authority System



HTTPS / TLS: When and How

Where should HTTPS be used at minimum?

**EVERYWHERE &
ALWAYS**

HTTPS configuration best practices:

- https://wiki.mozilla.org/Security/Server_Side_TLS#Recommended_configurations
- <https://www.owasp.org/index.php/O-Saft>
- <https://www.ssllabs.com/projects/best-practices/>

SSL / TLS Protocol Versions



SSL v2 Fully broken. Do not use!

SSL v3 Also fully broken from "POODLE"

TLS 1.0 "OK" but on its way out

TLS 1.1 No practical attacks; minimum for PCI

TLS 1.2 Solid well supported version

TLS 1.3 RFC 8446 published August 2018

TLS and June 30, 2018

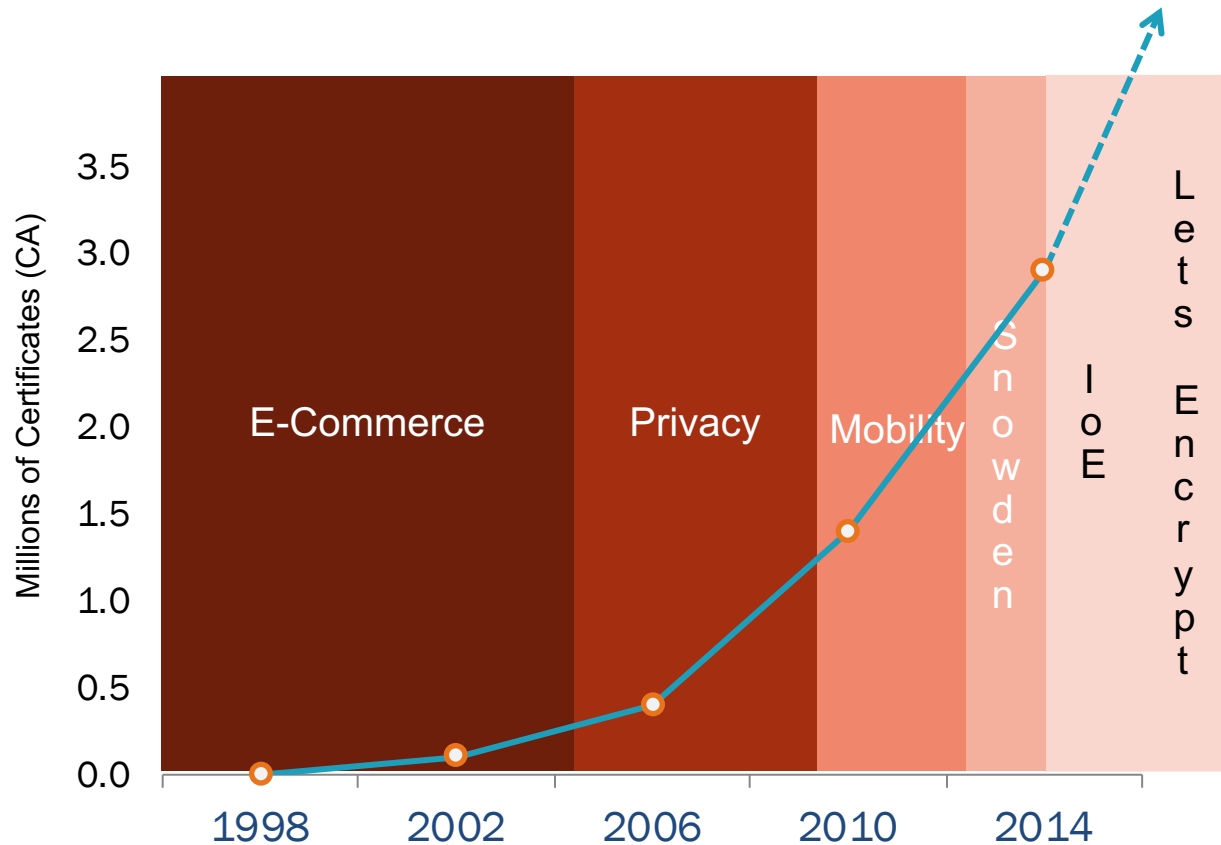


- June 30, 2018 is the deadline for enabling **TLS v1.1 or higher** in order to meet the PCI Data Security Standard (PCI DSS) for safeguarding credit card payment data
- TLS v1.2 or greater is strongly encouraged



Trajectory and Growth of Encryption

SSL growing ~30% annually. Well into the 5th wave of transition (IoE).



Source: Netcraft

As of February 2019 **48.2% of the web uses HTTPS by default**

<https://w3techs.com/technologies/details/ce-httpsdefault/all/all>

In a nutshell...

SSL stands for **Secure Sockets Layer (1994)**

TLS stands for **Transport Layer Security**
and is just the new name for SSL

- TLS 1.0 == SSL 3.1
- TLS 1.1 1999
- TLS 1.2 2008
- TLS 1.3 2017 Draft Status

HTTPS is **Hypertext
Transport Protocol Secure**
and provides HTTP
over SSL/TLS



TLS Protocol Workflow

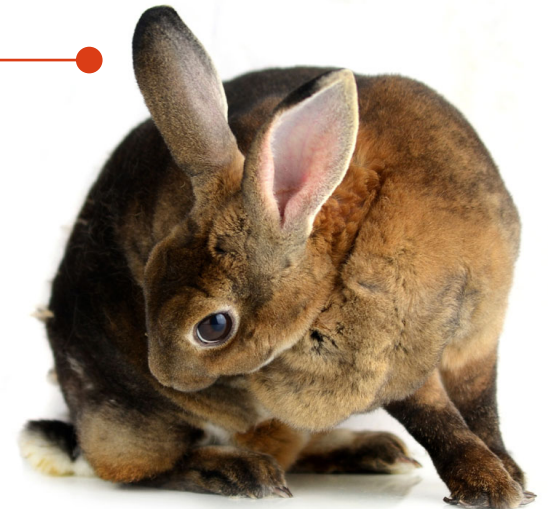
TLS uses both Symmetric and Asymmetric encryption

TLS exchanges Symmetric keys encrypted with Asymmetric keys, then falls back to Symmetric encryption

Why?

Symmetric encryption is MUCH faster

Asymmetric is slower but is used for authentication



TLS Certificates



TLS uses **X.509 Certificates**

Used to **authenticate the other party**

NOT used to help negotiate a symmetric key (beyond authentication)

TLS certificates from certificate authorities help websites **prove their authenticity.**

These certificates contain:

- The certificate holder
- The domain that the certificate was issued to
- The signature of the Certificate Authority who verified the certificate

Digital Signatures

Data Integrity and Message Authentication

Symmetric Key (single key) Integrity: Keyed-Hash MAC's (Message Authentication Codes)

- Does not separate read/write (i.e. verify/sign) privileges; the same key is used for both;
- Offers much better performance than digital signatures; and
- Requires much smaller key sizes than digital signatures.

Asymmetric Key (key pair) Integrity: Digital Signatures

- Separate read/write (i.e. verify/sign) privileges; a public key is used for one, alongside a private key for the other;
- Performance is much worse than message authentication codes; and
- Requires much larger key sizes than message authentication codes.

Digital Signature

A cryptographic primitive that ensures:

Data origin authentication of the signer

Integrity of the data

Non repudiation

Basic properties

Easy for the signer to sign data

Easy for everyone to verify the signature

Hard for anyone to forge a signature

Possible to have a third party settle any disputes

Confidentiality?

**Digital signatures do not
provide confidentiality!!**

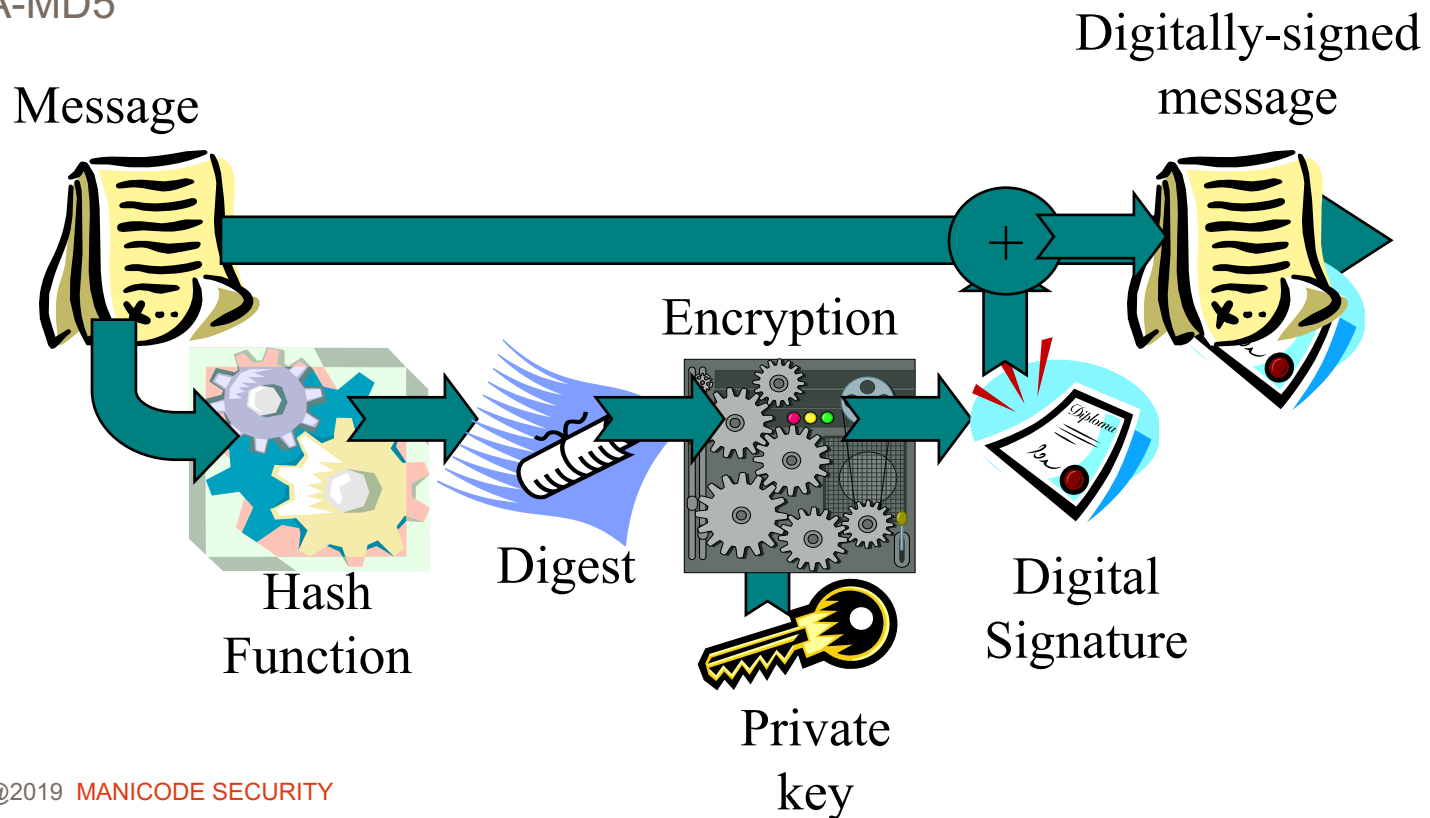
You will need to use some form of encryption for that

RSA digital signature scheme with appendix

Goal: authenticity and integrity

Usually named after the algorithms used.

E.g. RSA-MD5



Verifying a digital signature

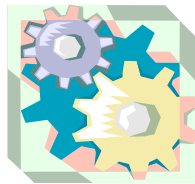
Digitally-signed
message



Message



Hash function



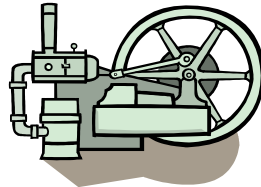
Digest



If identical, ok



Decryption



Digest'



Public
key

Why hash?

Integrity

An attacker can change a message (e.g. rearrange blocks)

An attacker might be able to even “sign” a forged message

Efficiency

RSA is a computationally expensive algorithm

Better to sign a small amount of data (hash) rather than the entire message

Relationship between CAs

Cross certification

Each CA certifies the other CA's public key.

CA Hierarchies

Introduce different levels of CAs

Certificate chain of trust

A series of different certificates that need to be verified in order to achieve trust end-to-end.

HTTPS Historical Problems

Other SSL Fails

- Posting passwords or other sensitive data over HTTP
- Loading mixed content
- Using weak version of SSL or TLS
- Using weak ciphers
- Terminating TLS early in your infrastructure
- Trusting the CA system
- Using old OSs or Webservers
- Old versions of OpenSSL



HTTPS / SSL Protocol #fail

2011 BEAST

- Upgrade to TLS 1.1
- Use RC4 for older protocols. Mitigated by browsers

2012 CRIME

- Stop using TLS compression

2013 BREACH

- Vendors disabled HTTP compression in client and server side
- Mask Sensitive tokens, randomize them per request

2014

- Heartbleed. Abuse the heartbeat feature in OpenSSL to retrieve chunks of memory for the server. (Not SSL's Problem)
- Death of SSL 3 (Poodle)

2015

- Superfish strikes Levono Machines
- Mozilla and Google revoke CNNIC root certs

2016

- SLOTH and DROWN attacks

2017

- Google and Mozilla begin plans to stop trusting existing Symantec certs



POODLE (Padding Oracle On Downgraded Legacy Encryption)

- **POODLE** is a flaw that exploits the lack of padding verification in SSL3. TLS 1+ does, which is why POODLE doesn't affect it.
- Plausible attack scenario targets session cookie data
 - Attacker runs fake WIFI
 - Injects evil JavaScript into HTTP site user is visiting
 - Evil JS makes requests to HTTPS target application
 - Attacker causes TLS failure triggering the use of SSL 3.0
 - Uses Oracle Padding attack (like CRIME and BEAST)
 - Victims session cookie is discovered byte by byte



Disable SSL 3.0 for all servers and clients

Use `TLS_FALLBACK_SCSV`

HTTPS / TLS Browser #fail



The site's security certificate is not trusted!

You attempted to reach ██████████ but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.

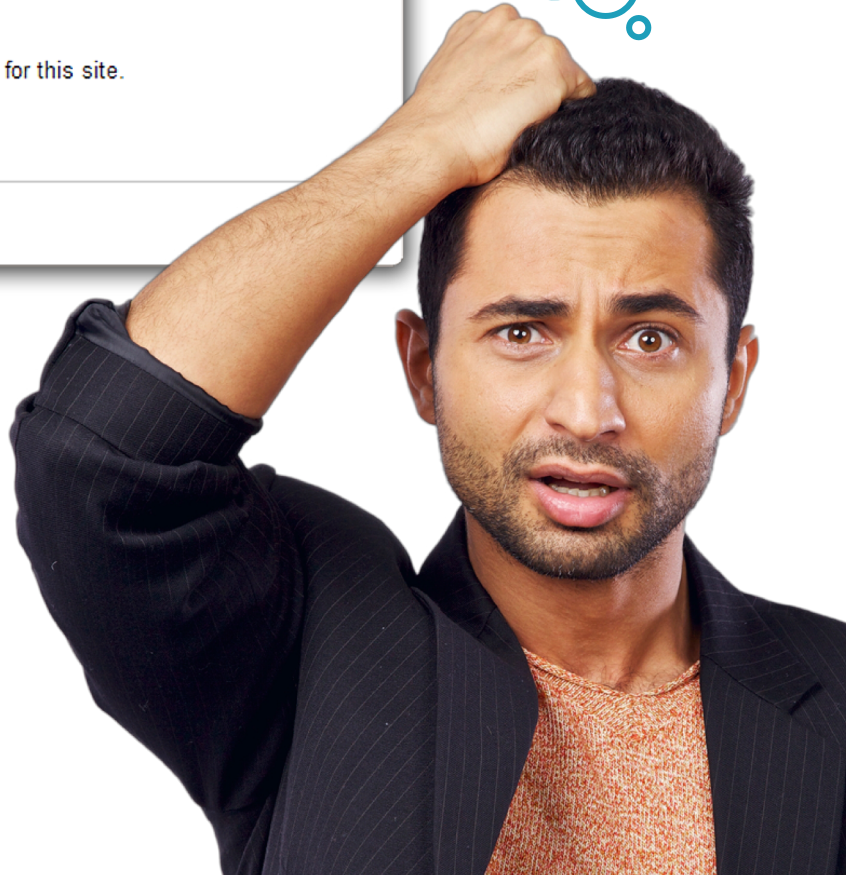
You should not proceed, **especially** if you have never seen this warning before for this site.

[▶ Help me understand](#)

It worked yesterday...

The effects of a “fail open/soft fail” policy

- 30-70% of the users click through warnings*
- Completely defeats the purpose of encryption
- No good way to change browser behavior, until recently



* <http://www.cs.berkeley.edu/~devdatta/papers/alice-in-warningland.pdf>

HTTPS / TLS CA #fail: September 2011

DigiNotar dies from certificate hack caper

'Unlikely many are going to shed tears' over Dutch company's demise, says security researcher



By Gregg Keizer

FOLLOW

Computerworld | Sep 21, 2011 5:09 PM PT

The Dutch company that was hacked earlier this summer by certificate thieves has gone bust and shut down, its U.S.-based owner said Tuesday.

DigiNotar filed for bankruptcy in a Netherland court on Monday, and its assets will be liquidated by a court-appointed trustee, said Vasco Data Security International, the Chicago company that purchased DigiNotar last January for \$13.1 million.

MORE LIKE THIS

Hackers stole Google SSL certificate, Dutch firm admits

Hackers steal SSL certificates for CIA, MI6, Mossad

Hackers may have stolen over 200 SSL certificates

on IDG Answers →

Is Microsoft engaging in dirty tricks with Windows 8's boot UEFI?

HTTPS / TLS CA #fail: February 2012

Home > Internet

News

Trustwave admits issuing man-in-the-middle digital certificate; Mozilla debates punishment

The issuing of subordinate root certificates to companies, so they can snoop on SSL-encrypted traffic, is a common industry practice

By Lucian Constantin

February 8, 2012 02:41 PM ET  Add a comment



IDG News Service - Digital Certificate Authority (CA) Trustwave revealed that it has issued a digital certificate that enabled an unnamed private company to spy on SSL-protected connections within its corporate network, an action that prompted the Mozilla community to debate whether the CA's root certificate should be removed from Firefox.

The certificate issued by Trustwave is known as a subordinate root and enabled its owner to sign digital certificates for virtually any domain on the Internet. The certificate was to be used within a private network within a data loss prevention system, Trustwave said in [a blog post](#) on Saturday.

HTTPS / TLS CA #fail: December 2012

Turkish Certificate Authority screwup leads to attempted Google impersonation

by Chester Wisniewski on January 4, 2013 | 2 Comments

FILED UNDER: [Apple Safari](#), [Featured](#), [Firefox](#), [Google](#), [Google Chrome](#), [Internet Explorer](#), [Privacy](#), [Vulnerability](#), [Web Browsers](#)

Some days I feel like a [broken record](#) that keeps [repeating](#) the [same story](#). Today's repeat meme? Certificate Authority issues certificates leading to Google users being spied upon.

Google first detected the problem, using Chrome's [certificate pinning](#), on December 24th, according to a blog post titled "[Enhancing digital certificate security](#)."

That title is a bit misleading, as this notification was not at all about proactively enhancing the safety of digital certificates, rather more of a mopping up from a large accident.

An unknown Google Chrome user was alerted to a validly signed Google certificate that did not in fact belong to Google.



HTTPS / TLS CA #fail: December 2013

Home > Security

News

Other browser makers follow Google's lead, revoke rogue certificates

Microsoft, Mozilla and Opera nullify unauthorized French certificates; Windows XP users out of luck

By Gregg Keizer

December 9, 2013 03:56 PM ET  Add a comment



Computerworld - Microsoft, Mozilla and Opera Software today joined Google in revoking rogue digital certificates that had been issued by a subordinate certificate authority (CA) of France's cybersecurity agency.

[Google revoked the certificates](#) for users of its Chrome browser on Saturday after a four-day investigation. Microsoft, Mozilla and Opera Software followed suit on Monday.

Browser / OS TLS SINS: February 2014

Vulnerability Summary for CVE-2014-1266

Original release date: 02/22/2014

Last revised: 02/24/2014

Source: US-CERT/NIST

Overview

The SSLVerifySignedServerKeyExchange function in libsecurity_ssl/lib/sslKeyExchange.c in the Secure Transport feature in the Data Security component in Apple iOS 6.x before 6.1.6 and 7.x before 7.0.6, Apple TV 6.x before 6.0.2, and Apple OS X 10.9.x before 10.9.2 does not check the signature in a TLS Server Key Exchange message, which allows man-in-the-middle attackers to spoof SSL servers by (1) using an arbitrary private key for the signing step or (2) omitting the signing step.

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 6.8 (MEDIUM) (AV:N/AC:M/Au:N/C:P/I:P/A:P) (legend)

Impact Subscore: 6.4

Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

Access Vector: Network exploitable

Access Complexity: Medium

Authentication: Not required to exploit

Impact Type: Allows unauthorized disclosure of information; Allows unauthorized modification; Allows disruption of service

Apple goto #fail SSL / TLS bug

Major iOS/OSX SSL implementation bug

<http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-1266>

"...does not check the signature in a TLS Server Key Exchange message...."

"...allows man-in-the-middle attackers to spoof SSL servers by (1) using an arbitrary private key for the signing step or (2) omitting the signing step."

"goto fail" Apple SSL / TLS bug



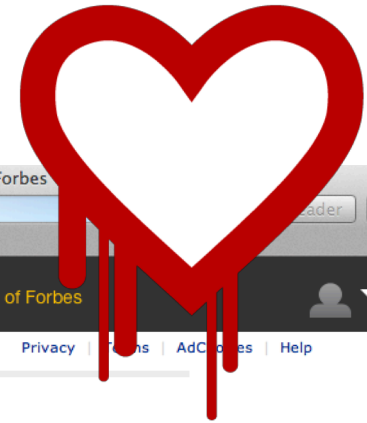
← → ↻ view-source

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer
    signedParams, uint8_t *signature, UInt16 signatureLen)
{
    OSStatus          err;
    ...

    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;

    ...
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

OpenSSL Sins: Heartbleed 2014



The screenshot shows a web browser window displaying a Forbes article. The browser's address bar shows the URL: www.forbes.com/sites/jameslyne/2014/04/08/heartbeat-heartbleed-bug-breaks-worldwide-internet-security-again-and-yahoo/. The Forbes logo is in the top left, and navigation links like 'New Posts +35', 'Most Popular', 'Lists', 'Video', and '2 Free Issues of Forbes' are in the top right. The article title is 'Heartbeat Heartbleed Bug Breaks Worldwide Internet Security Again (And Yahoo)'. The author is James Lyne, a contributor. The article text discusses the Heartbleed bug, its impact on internet security, and mentions that Yahoo is no longer vulnerable to the attack but may have had significant data leaked during the vulnerable period.

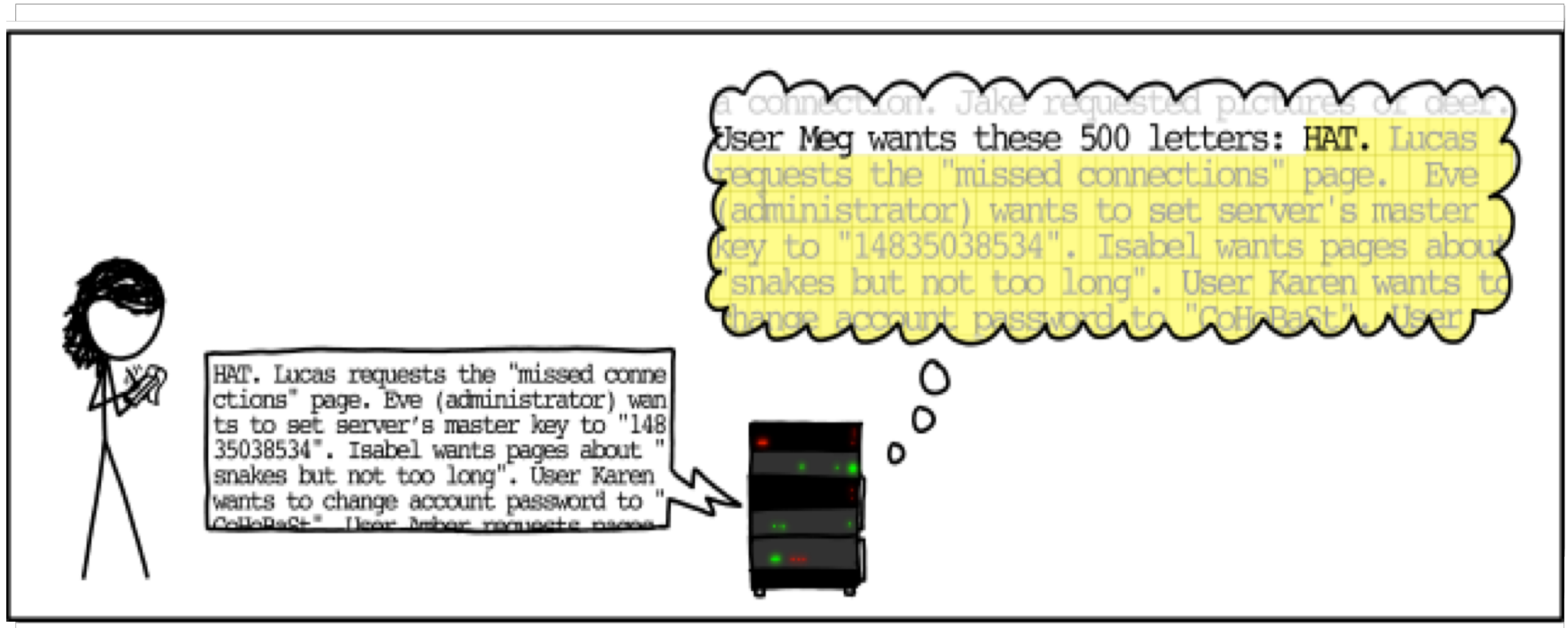
Heartbeat Heartbleed Bug Breaks Worldwide Internet Security Again (And Yahoo)

TECH 4/08/2014 @ 11:39AM | 103,033 views

[+ Comment Now](#) [+ Follow Comments](#)

Every day you use [encryption](#) technology to protect your data, your applications and online services . Most of the time most people are blissfully unaware it is even happening. Whether you are a consumer accessing your Internet bank site, using a mobile application to log in and share data or trading online most of our use of modern technology involves this key capability and without it trust on the Internet is significant undermined. A new bug, again, puts trust on the Internet at risk on a significant scale. The bug, dubbed 'heartbleed' is based on a fault in functionality in the widely used [OpenSSL](#) library. It was originally discovered by Neel Mehta of [Google](#) Security. This library is extremely widely used from security vendors products to secure web browsing (when you log in to a site and see [https://](#)) and even mobile banking applications. The [Apache web server](#) which powers a substantial part of the Internet tends towards using OpenSSL. You may be using it at your business right now and many popular services like [Yahoo](#) have been shown to be vulnerable (see the image below). **UPDATE** *Yahoo is no longer vulnerable to the attack, but there may have been significant data leaked for the extended period where they were running the vulnerable software.*

How Heartbleed works



Authored by XKCD!

Chrome Revokes Symantec's Authority

- Symantec once owned and ran four major certificate authorities
 - Symantec
 - GeoTrust
 - Thawte
 - RapidSSL
- All Symantec certificates are being revoked that were created on Symantec's older certificate creation infrastructure
- Revocation Timeline
 - **October 2017/Chrome 62** Chrome DevTools will start alerting when receiving certificates from Symantec that will be revoked in Chrome 66.
 - **April 2018/Chrome 66** All Symantec certificates created before June 1, 2016 will no longer be trusted by Chrome (ie: old Symantec infrastructure)
 - **October 2018/Chrome 70** All symantec certificates created on Symantecs older infrastructure no longer be trusted by Chrome

SHARE



SHARE



TWEET



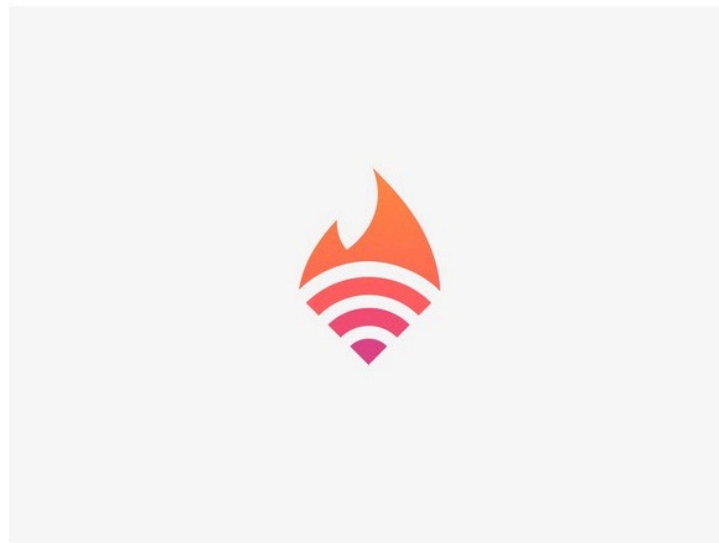
COMMENT



EMAIL

ANDY GREENBERG SECURITY 01.23.18 09:00 AM

TINDER'S LACK OF ENCRYPTION LETS STRANGERS SPY ON YOUR SWIPES



MAI SCHOTZ



MOST POPULAR



GEAR
Gboard Is the Best Keyboard For Most Smartphones
BRIAN BARRETT



SCIENCE
Yes, There Is Gravity in Space
RHETT ALLAIN



SCIENCE
3.5 Billion-Year-Old Fossils Challenge Ideas About Earth's Start
REBECCA BOYLE



MORE STORIES








HTTPS Progress

Chrome 62 in October 2017

Will mark ALL Incognito HTTP sites as insecure

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

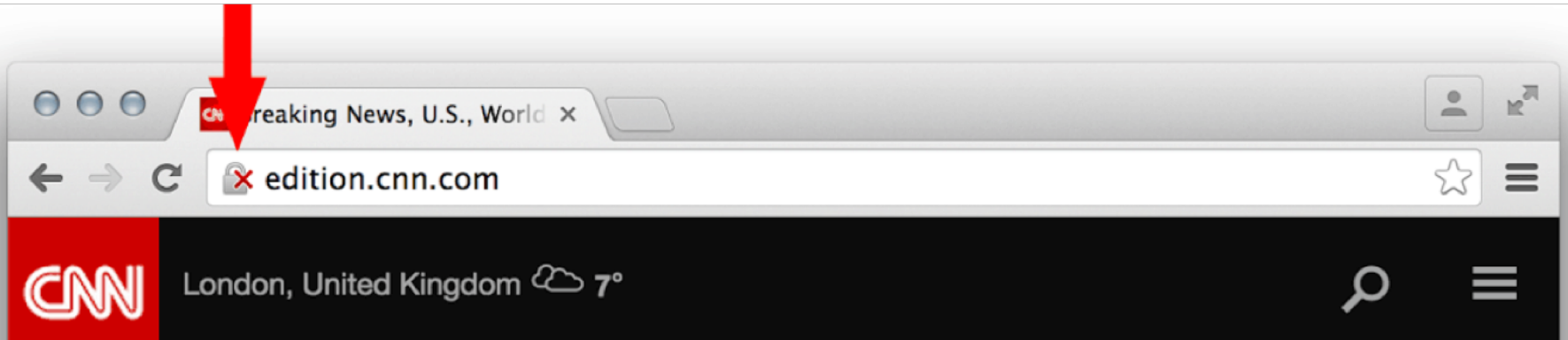
(Chrome 53)	 login.example.com
Jan. 2017 (Chrome 56)	 Not secure login.example.com

	Treatment of HTTP pages outside Incognito mode:	 Treatment of HTTP pages in Chrome Incognito mode:
(Chrome 58)	 example.com	 example.com
Oct. 2017 (Chrome 62) at page load	 example.com	 Not secure example.com
Oct. 2017 (Chrome 62) when entering data	 Not secure example.com	 Not secure example.com

Chrome mark HTTP sites as insecure

HTTP is downgraded as insecure in Chrome 68+

As of July 24 2018 Chrome release 68 marks all HTTP sites as insecure



Chrome 68 July 24, 2018

ALL HTTP sites marked as insecure



Excuses to avoid supporting HTTPS

Aren't certificates expensive/difficult to obtain?

The "Let's Encrypt" project will make it easy to obtain free certs for as many (sub)domains as desired!

Isn't SSL/TLS slow?

<https://istlsfastyet.com/>

Doesn't HTTPS break caching? Filtering?

For environments that need tight control of internet access, there are several client-side/network solutions.

<https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure>

My site not that important

HTTPS is no longer just for security. It's a critical part of user experience. (Geolocation, AppCache, getUserMedia() and PUSH notifications)



thegameoffitness.com

Improving HTTPS

HSTS (Strict Transport Security)

http://www.youtube.com/watch?v=zEV3HOuM_Vw

Strict-Transport-Security: max-age=31536000

Forward Secrecy

<https://whispersystems.org/blog/asynchronous-security/>

Mozilla Recommended TLS Security Configurations

https://wiki.mozilla.org/Security/Server_Side_TLS#Recommended_configurations

Mozilla SSL Configuration Generator

<https://mozilla.github.io/server-side-tls/ssl-config-generator/>

NIST Guidelines on selection, configuration and use of TLS

https://www.nist.gov/publications/guidelines-selection-configuration-and-use-transport-layer-security-tls-implementations?pub_id=915295

Certificate Pinning

https://www.owasp.org/index.php/Pinning_Cheat_Sheet

HSTS

HTTP Strict Transport Security

About HSTS

- Released in November 2012
- Mitigates
 - Downgrade to HTTP attacks
 - MitM attack using DNS trickery
 - Browser default behavior of trying HTTP first
 - Mixed content
- **Protects the user**, not the website
- HTTPS specific and must own the domain

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

HTTP Strict Transport Security (HSTS)

HSTS (Strict Transport Security)

http://www.youtube.com/watch?v=zEV3HOuM_Vw

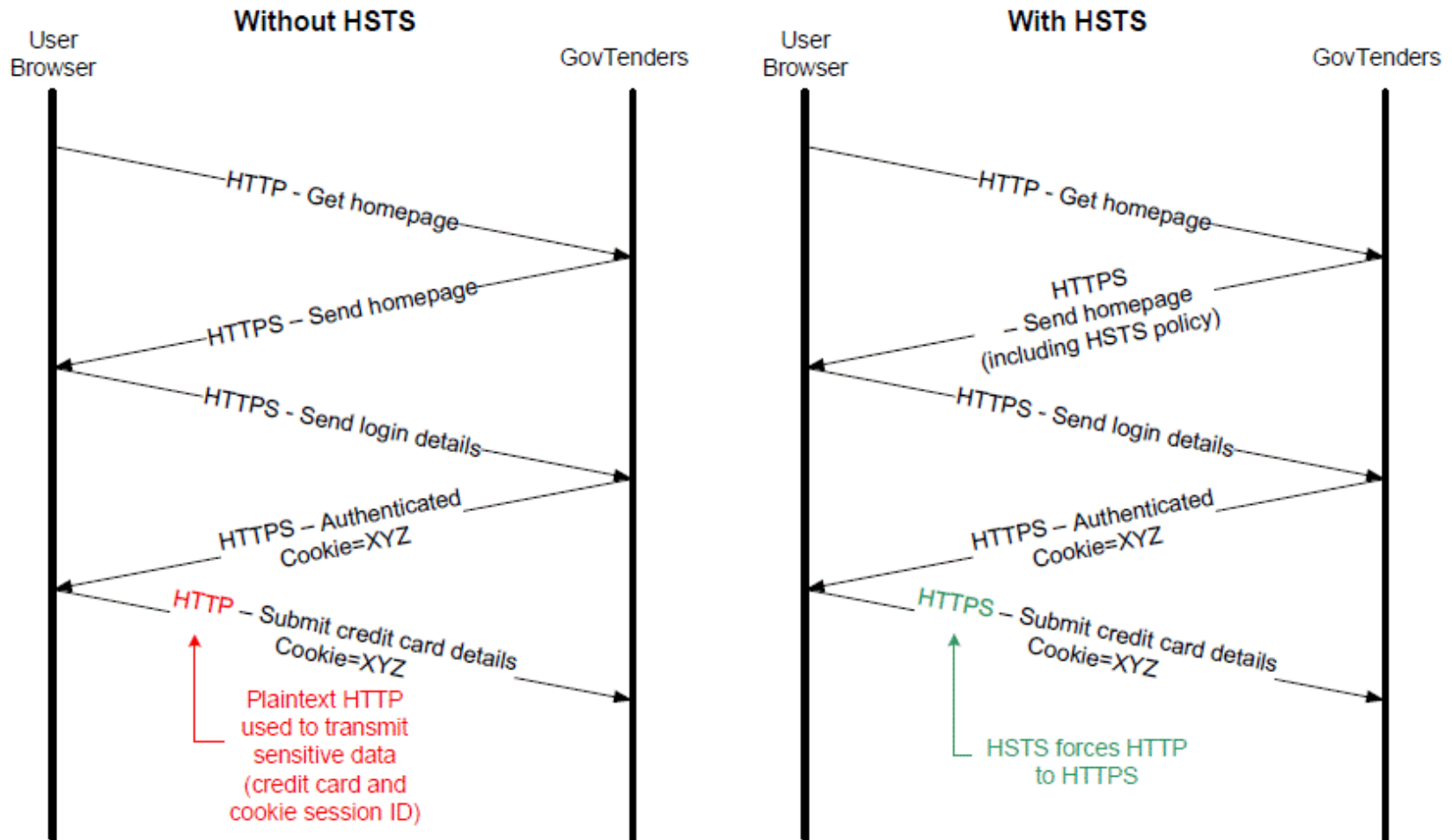
Strict-Transport-Security:
max-age=31536000; includeSubdomains;

Forces browser to only make
HTTPS connection to server

Must be initially delivered
over a HTTPS connection



Example of HSTS in action



Reference: http://www.asd.gov.au/publications/protect/protecting_web_apps.htm

HSTS: Preload list

- To add your site that you would like to see included in the preloaded HSTS list visit: <https://hstspreload.org/>
- Current HSTS Chrome preload list: https://chromium.googlesource.com/chromium/src/+master/net/http/transport_security_state_static.json
- A site is included in the Firefox preload list if the following hold:
 - Serve a valid certificate.
 - Redirect from HTTP to HTTPS on the same host, if you are listening on port 80.
 - Serve all subdomains over HTTPS.
 - The max-age must be at least 31536000 seconds (1 year).
 - The includeSubDomains directive must be specified.
 - The preload directive must be specified.
 - If you are serving an additional redirect from your HTTPS site, that redirect must still have the HSTS header.
- Need to remove yourself from the list?
<https://hstspreload.org/removal/>

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

HSTS Tips...

Won't work with **IP**

Won't work with **self signed certs**

Won't work with **plaintext connection**

Will work with **all ports**

Deploy with a **short duration** value first. Increase it later.

To revoke HSTS:

```
Strict-Transport-Security: max-age=0
```

HSTS Security Considerations

- **Include in all subdomains**, even when using `includeSubDomains`
 - Lack of `includeSubDomains` is a privacy violation for users.
- **Back to first use scenario once retention period expires.**
 - Can be forced to first use scenario by spoofing NTP
- **Does not necessarily secure cookies**
 - Redirect to a made-up subdomain may reveal cookies (assuming no https)
 - Continue to use secure cookies



HSTS Supported Browsers

Minimum Browser Support

Internet Explorer 11 Or Edge

Firefox 29

Opera 12

Safari 7

Android Browser 4.4 (KitKat)

Chrome

Perfect Forward Secrecy

Perfect Forward Secrecy

- If you use older SSL ciphers, every time anyone makes a SSL connection to your server, that message is encrypted with (basically) the same private server key
- **Perfect forward secrecy:** Peers in a conversation instead negotiate secrets through an ephemeral (temporary) key exchange
- With PFS, recording ciphertext traffic doesn't help an attacker even if the private server key is stolen!

<https://whispersystems.org/blog/asynchronous-security/>



SSL / TLS Ciphers

Forward Secrecy

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)

NOT Forward Secrecy

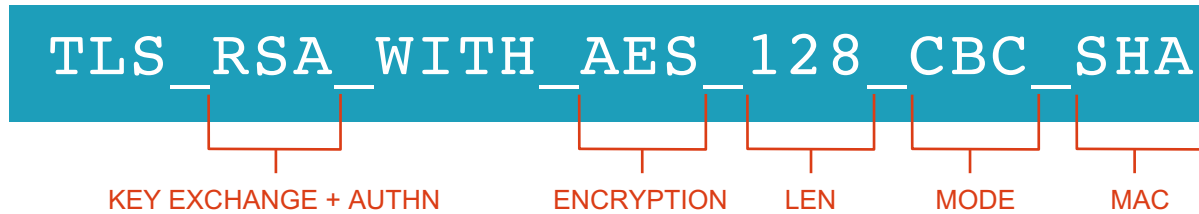
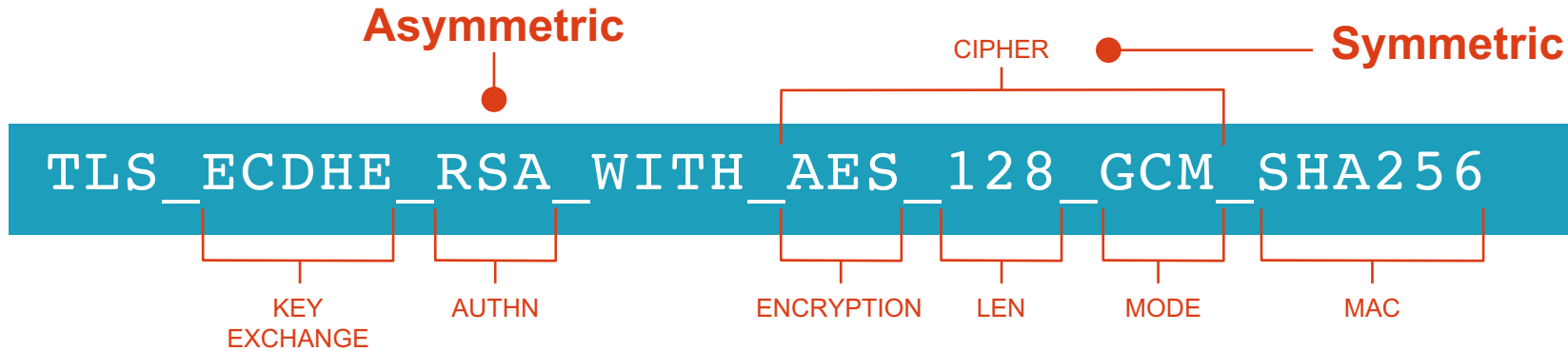
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)

TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)

TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)

TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)

Cipher Suite Explained



KX	AuthN	Encryption	MAC	PRF
RSA ★ ↓	RSA ★	AES ★ ↑	SHA256 ↑	SHA256 ★
ECDHE ↑	ECDSA ↑	RC4 ↓	SHA1	SHA384
DHE ↓	DSS	3DES ↓	MD5	Protocol*

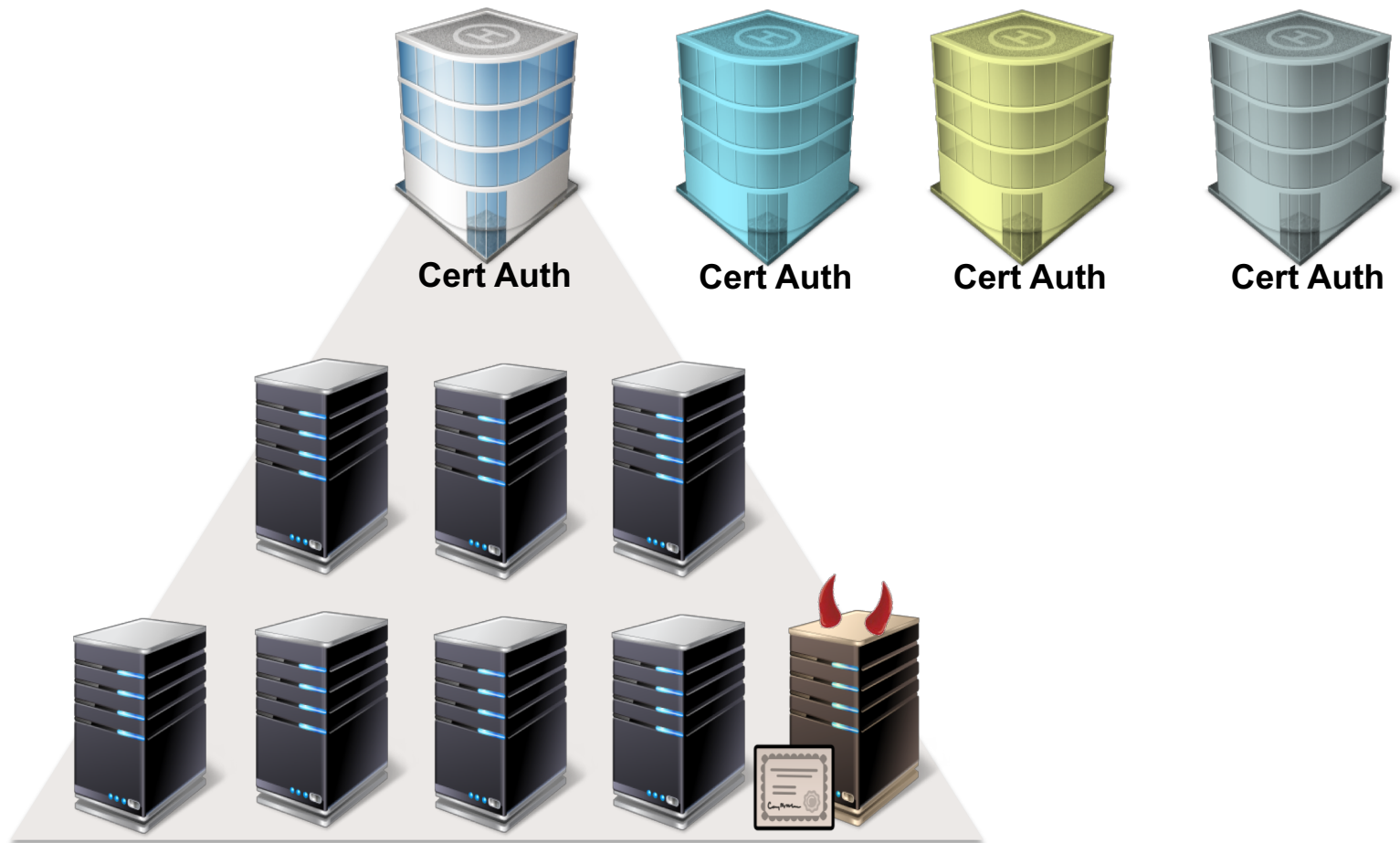
LEGEND

- ★ Popular
- ↓ Phase Out
- ↑ NIST std

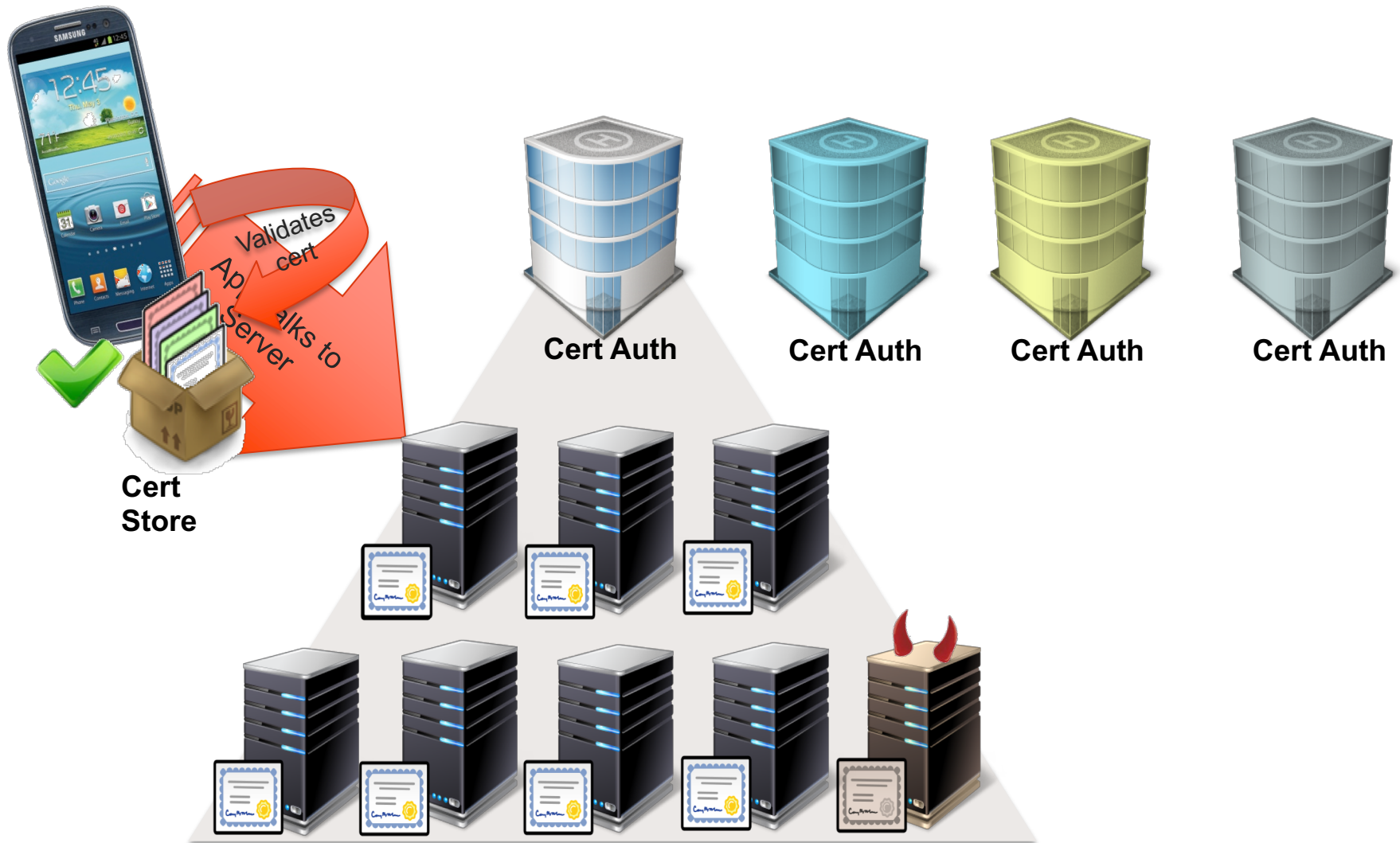
GCM	↑
CBC	↓

Mobile Certificate Pinning

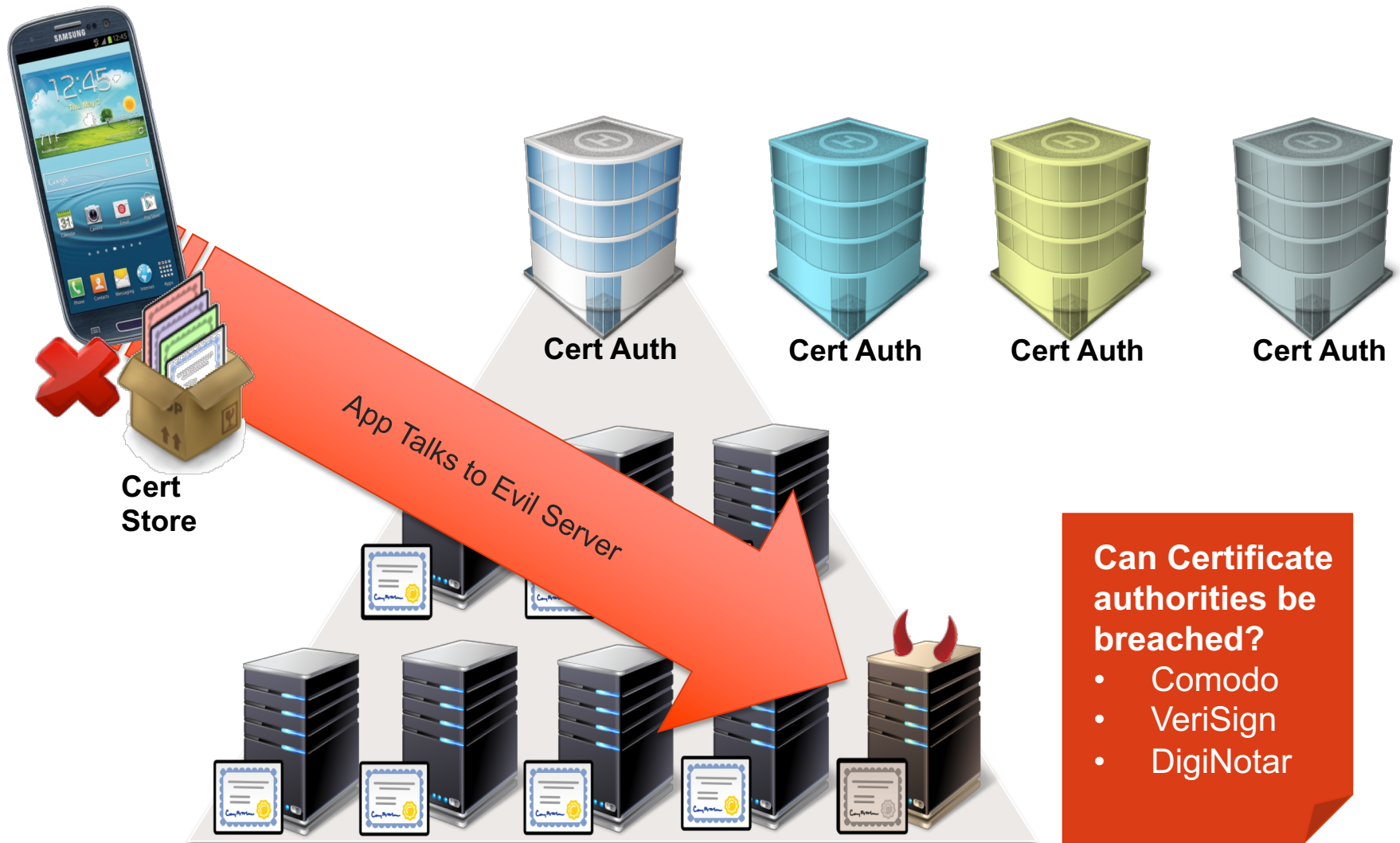
Certificate Authorities In ~~God~~ We Trust



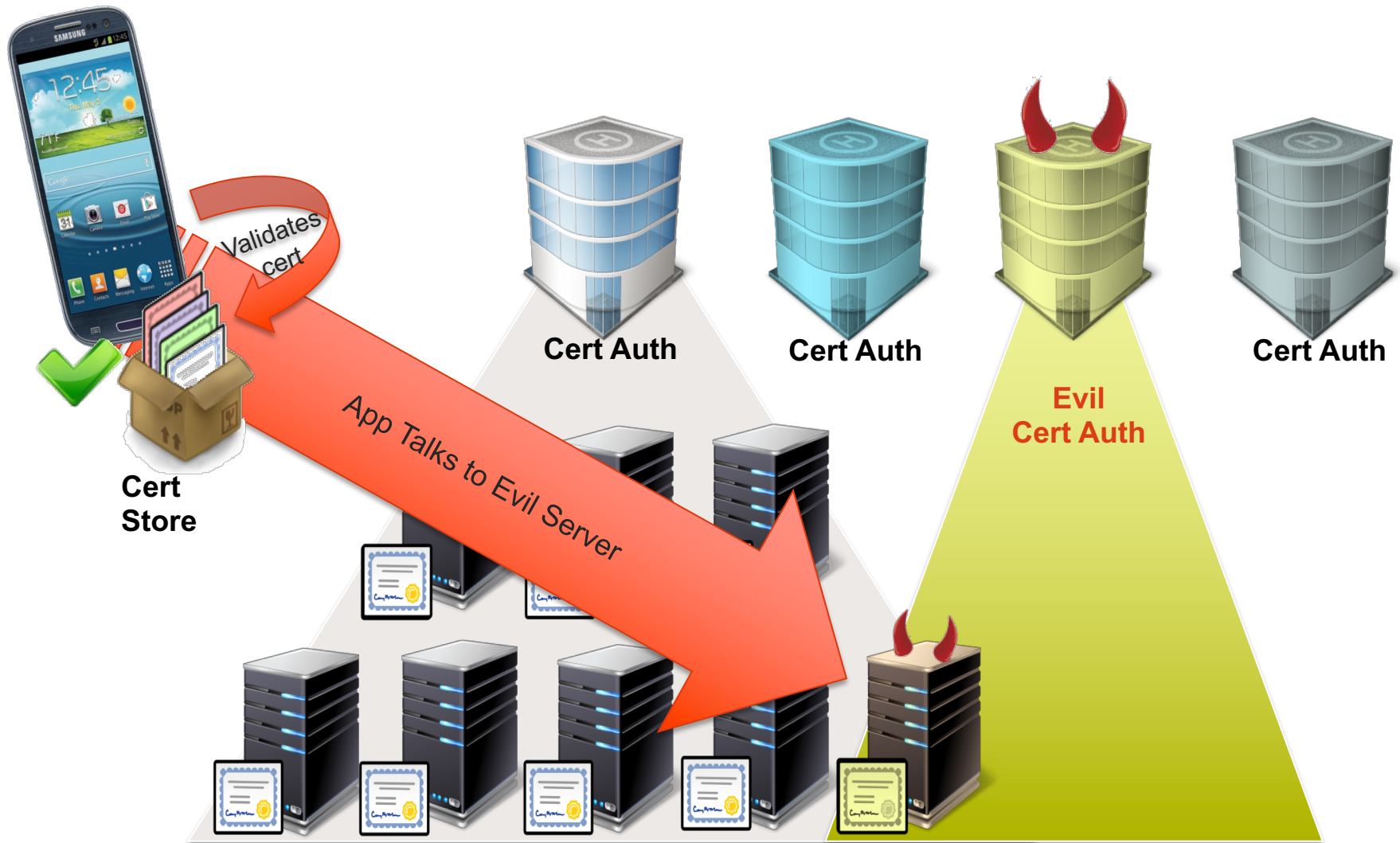
Certificate Authorities In ~~God~~ We Trust



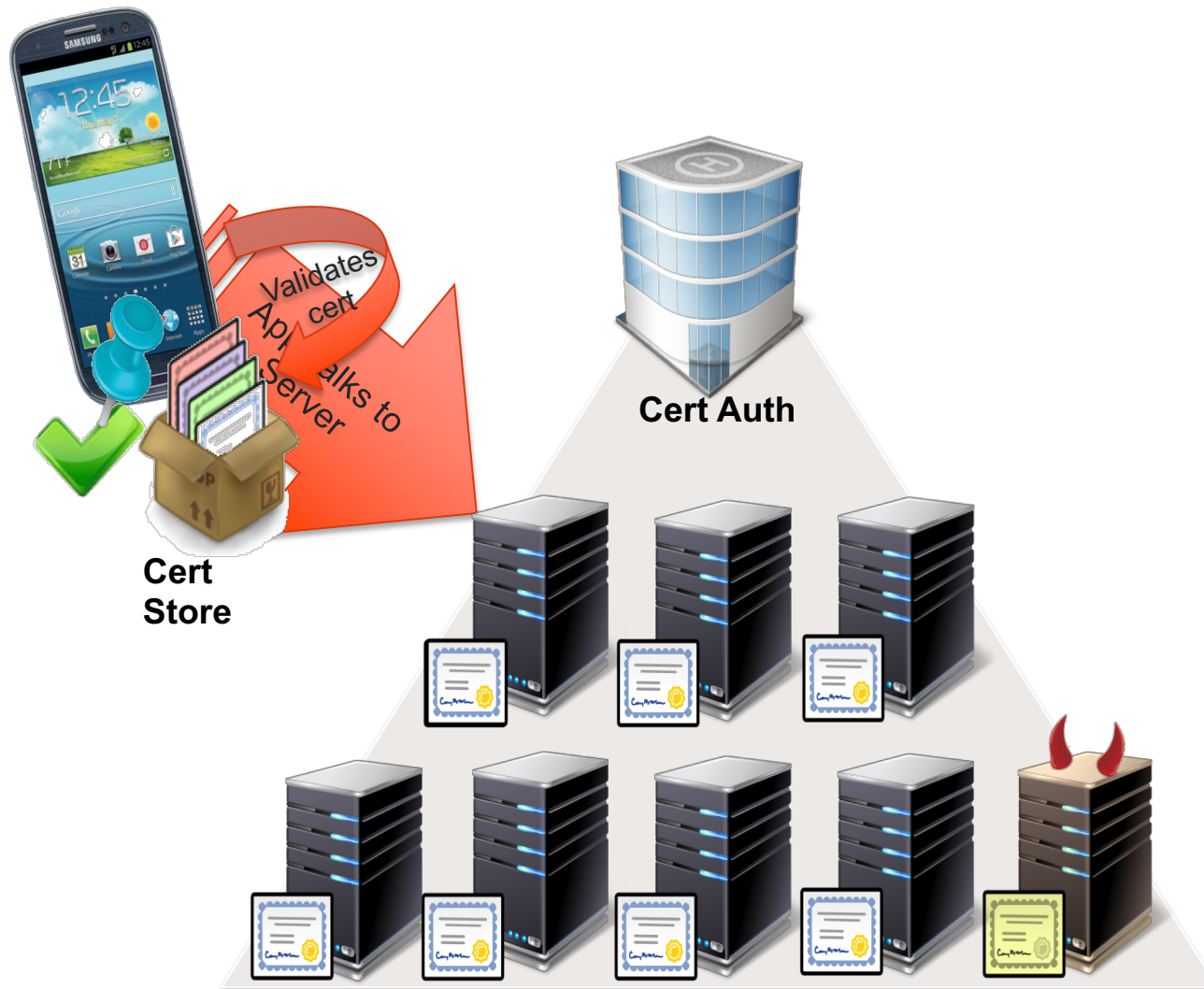
Certificate Authorities In ~~God~~ We Trust



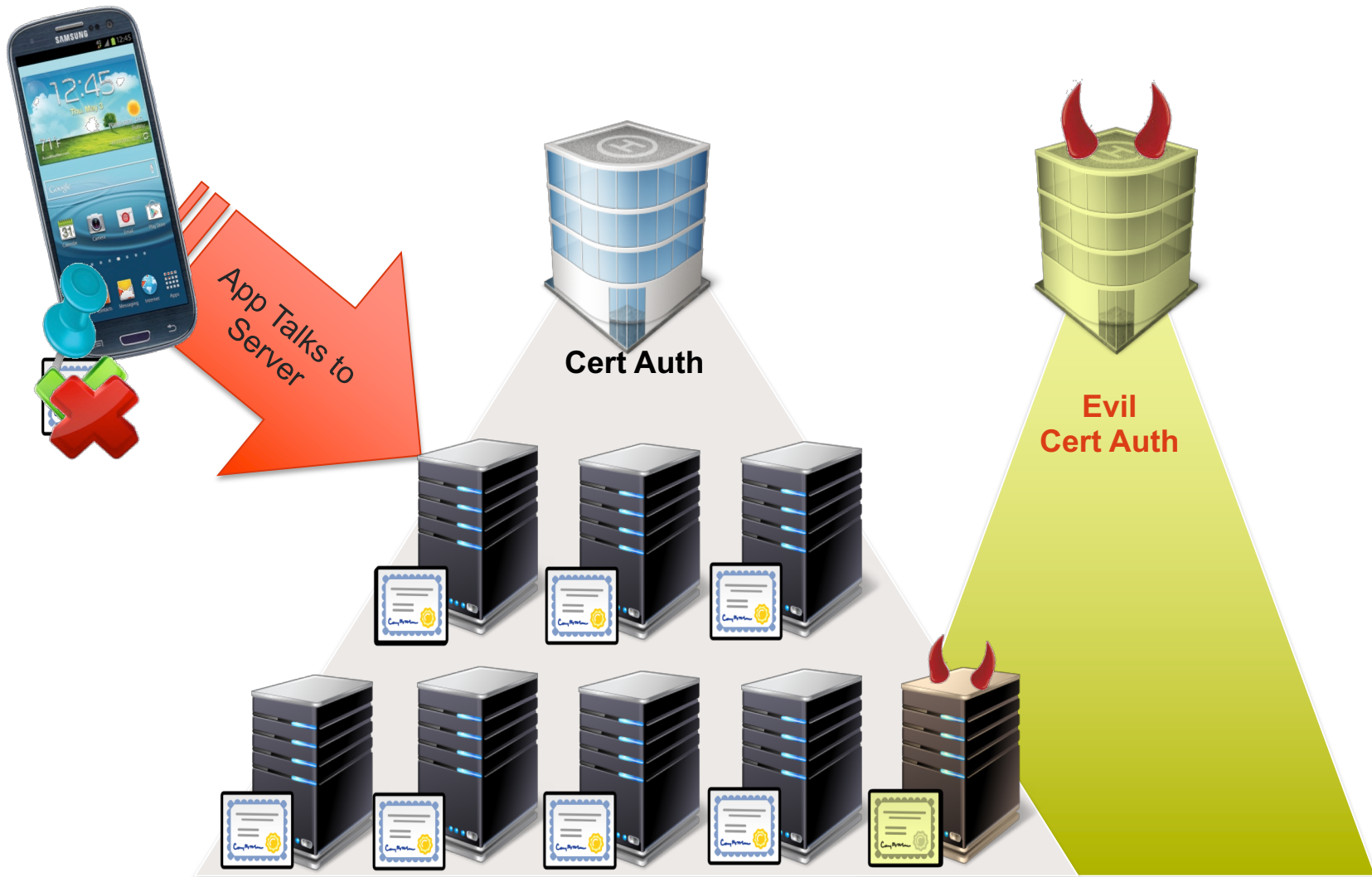
In God We Trust and maybe some Certificate Authorities...



Pinning: Trust On First Use (TOFU) example



Pinning: Reduces the attack surface for cert forgery



What is Certificate Pinning?

Pinning is a key continuity scheme, detecting when an imposter with a fake, but properly CA signed certificate attempts to act like the real server.

There are two types of pinning:

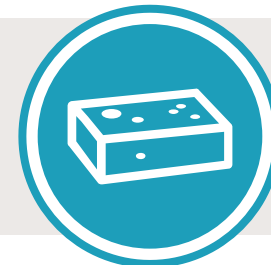
Carry around a copy of the server's public key

Great if you are distributing a dedicated client-server application since you know the server's certificate or public key in advance



Note of the server's public key on first use; Trust-on-First-Use (TOFU) pinning

Useful when no a priori knowledge exists, such as SSH or a Browser.



https://www.owasp.org/index.php/Pinning_Cheat_Sheet

Pinning Options: What to Pin

Certificate Pinning: Intermediate cert, Root cert

Public Key Pinning (may cover more certs):
subjectPublicKeyInfo, RSAPublicKey, DSAPublicKey

Hash of the options above

Pinning: Considerations

Always have a backup pin and a spare certificate from a different CA

- Avoids a self inflicted DoS
- Consider setting the pin in days if trust on first use (TOFU, aka key continuity)
- Certificates should have overlapping validity periods

For Mobile Apps, consider adding the pinning during install instead of TOFU

- Out of band pinning decreases chances of attacker tainting pin

Pinning makes it harder to Pen Test

- May need to use iOS SSL/TLS Kill Switch and Android SSL/TLS Bypass Tool
- May need to disable it altogether

Privately (locally installed) Certificate Authorities

"...private trust anchors can be used to proxy (or MITM) connections, even to pinned sites. 'Data loss prevention' appliances, firewalls, content filters, and malware can use this feature to defeat the protections of key pinning..."

<https://www.chromium.org/Home/chromium-security/security-faq#TOC-How-does-key-pinning-interact-with-local-proxies-and-filters->

Users can be...

- Tricked into installing a CA. (ex/ to get WiFi access at a hotel)
- Forced into installing a CA as part of a BYOD program.
- Forced into installing a CA just by installing a program.
- Forced into using a CA that is preloaded by an OEM.

Pinning Can (and has) Been Evaded

Locally Installed Authorities can MITM while evading Pinning!

"Researchers revealed that a vulnerability in Superfish software, which came pre-loaded on many Lenovo laptops, could let hackers impersonate shopping, banking and other websites and steal users' credit card numbers and other personal data."

— CBS News, February 2017

Even the hard-coded Pinning of Google services in Chrome were evaded

- SuperFish installs its own root CA certificate in Windows systems.
- It then generates certificates on the fly for each attempted SSL connection to inject advertisements.
- The private key of the Superfish pair was discovered allowing reuse.

Rapid research on this incident from the security community

- Robert Graham series of blog posts
- General Info on Superfish: <http://blog.erratasec.com/2017/02/some-notes-on-superfish.htm>
- Extracting Superfish certs: <http://blog.erratasec.com/2017/02/extracting-superfish-certificate.html>
- Exploiting Superfish: <http://blog.erratasec.com/2017/02/exploiting-superfish-certificate.html>

Android Sins: No clear way to remove local authority



android

Android Open Source Project - Issue Tracker

 Search projects

[Project Home](#)

Issues

[New issue](#)

Search

Open issues



for

Search

[Advanced search](#)

[Search tips](#)

[Subscriptions](#)

Issue [174714](#): No general purpose method to remove a CA certificate that was programitcally installed

2 people starred this issue and may be notified of changes.

[Back to lis](#)

Status: New

Owner: [kr...@android.com](#)

Cc: [kr...@android.com](#),
[b...@google.com](#),
[klyu...@google.com](#)

Type-Defect

Priority-Small

ReportedBy-User

[Sign in](#) to add a comment

Reported by [noloa...@gmail.com](#), May 26, 2015

This is an interesting issue from Stack Overflow. The question can be found at <http://stackoverflow.com/q/30327023>.

In the question, and app installed a CA. The CA was used for traffic inspection. When the app is going to be removed, the authors would like to remove the CA. It seems like a responsible thing to do for cleanup. (And no comment on the inspection).

However, it appears there's no general purpose method for the app to remove the CA it installed. In this case, they are trying to remove a CA they installed, and not other CAs in the system.

It seems to me, that if you are going to use the web/browser security model, embrace phishing and break known good pinsets (re: HPKP), then you should allow the phishers to remove their warez when they are done.


Recent Advances

Expect Certificate Transparency



- Provide an open auditing and monitoring system that lets any domain owner or CA determine whether certificates have been mistakenly or maliciously issued.
- 2015: Google required EV certs to use CT
- 2016: Google forces all new certs to use CT
- May 2018: Google forces ALL certs to use CT

Certificate Authority Authorization (CAA)

- Certificate Authority Authorization (CAA) is a standard that allows domain name owners to restrict which CAs are allowed to issue certificates for their domains.
- Certificate Authority Authorization (CAA) can help detect malicious issuance of fraudulent certificates. 
- **September 2017**: CAA became mandatory. CAs with browser root certificates must check for the CAA record and must follow the content of CAA when issuing new certificates.

Let's Encrypt

<https://letsencrypt.org/>

"free, automated, and open certificate authority (CA), run for the public's benefit"

[<https://letsencrypt.org/about/>]

Provided by ISRG - Internet Security Research Group (California based Non-Profit)

Sponsored by Mozilla, Akamai, Cisco, EFF, OVH,
Facebook, Chrome, etc.

Mission: reduce technological & financial barriers to
secure internet communication



Certificate revocation
doesn't always



Revocation does not work

It takes at least **10 days** for the revocation Information to fully *propagate*

Browser *soft fail policy* makes revocation *ineffective*

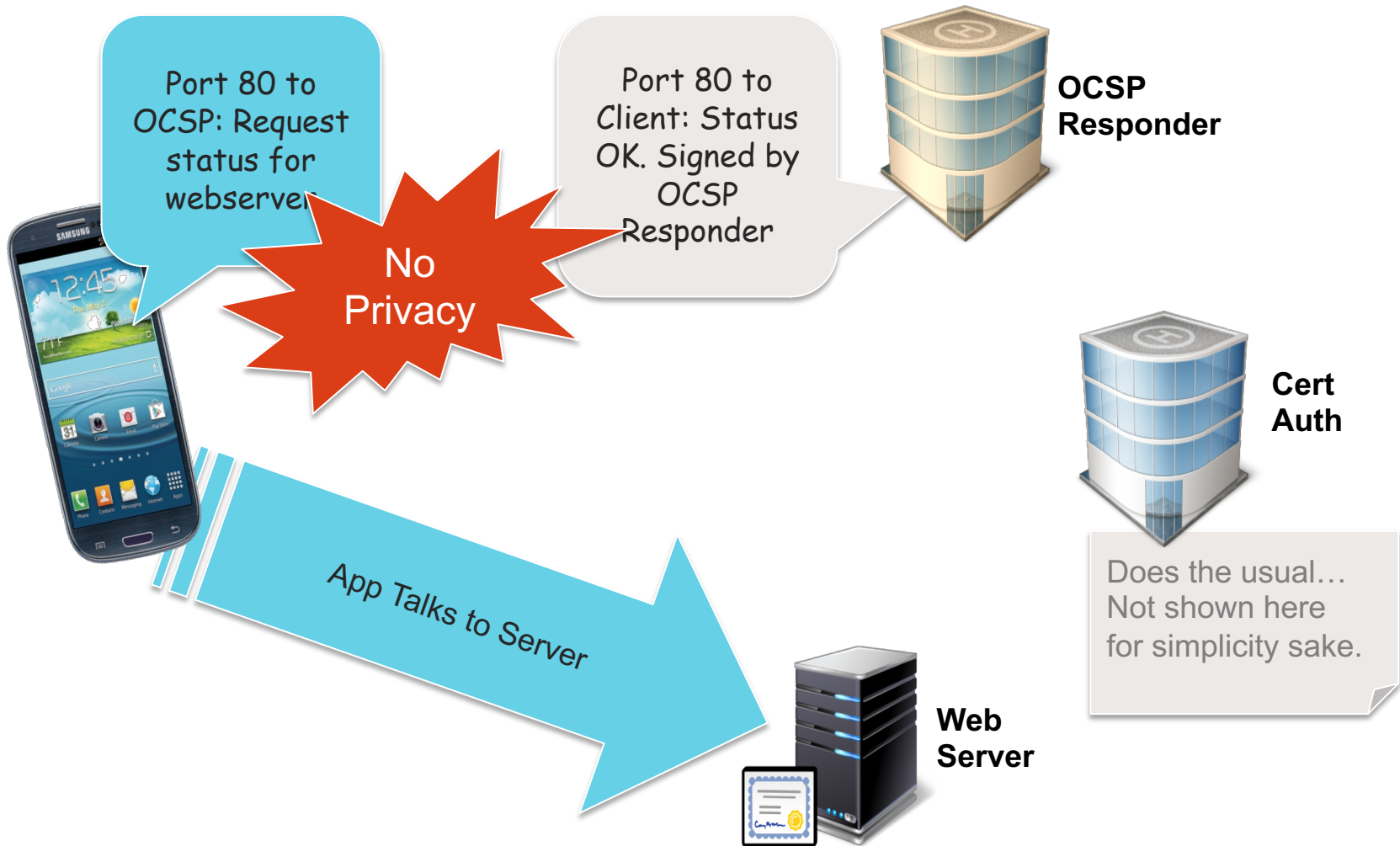
OCSP request can be *intercepted* (more on this than later)

Most browsers ignore revocation for all certificates but EV certificates

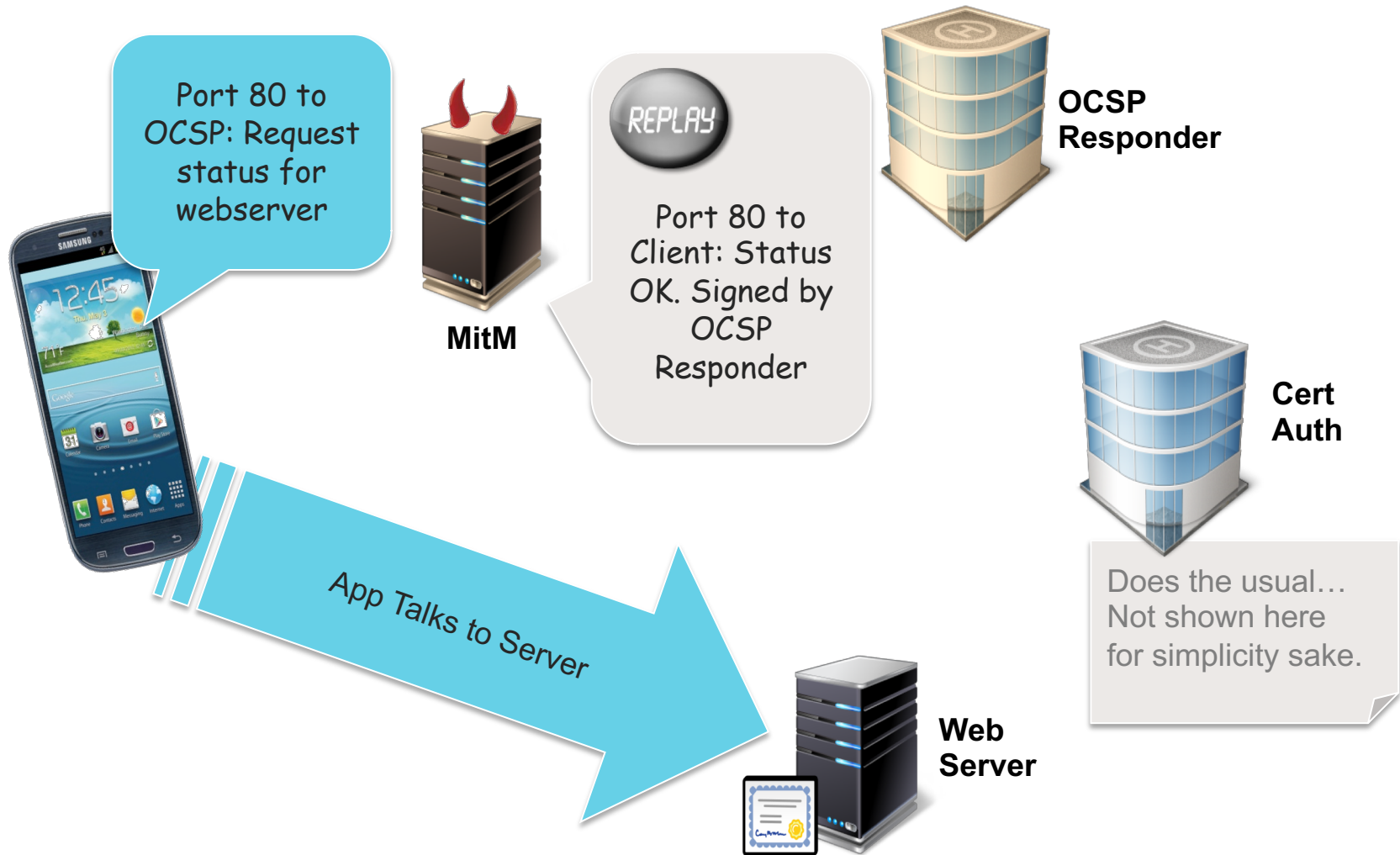
- List includes Chrome, Firefox, Chrome on Android, iOS on Safari...
IE and Opera do the right think checking OCSP and CRL when appropriate
- Revocation checks can be enable in some browsers.
In Firefox, set `security.OCSP.required` to `true`
- Important certificates (e.g., intermediate CAs), rely on a proprietary revocation channel (CRISets) that feeds off of Cert Revocation Lists (CRL) information

Online Certificate Status Protocol (OCSP)

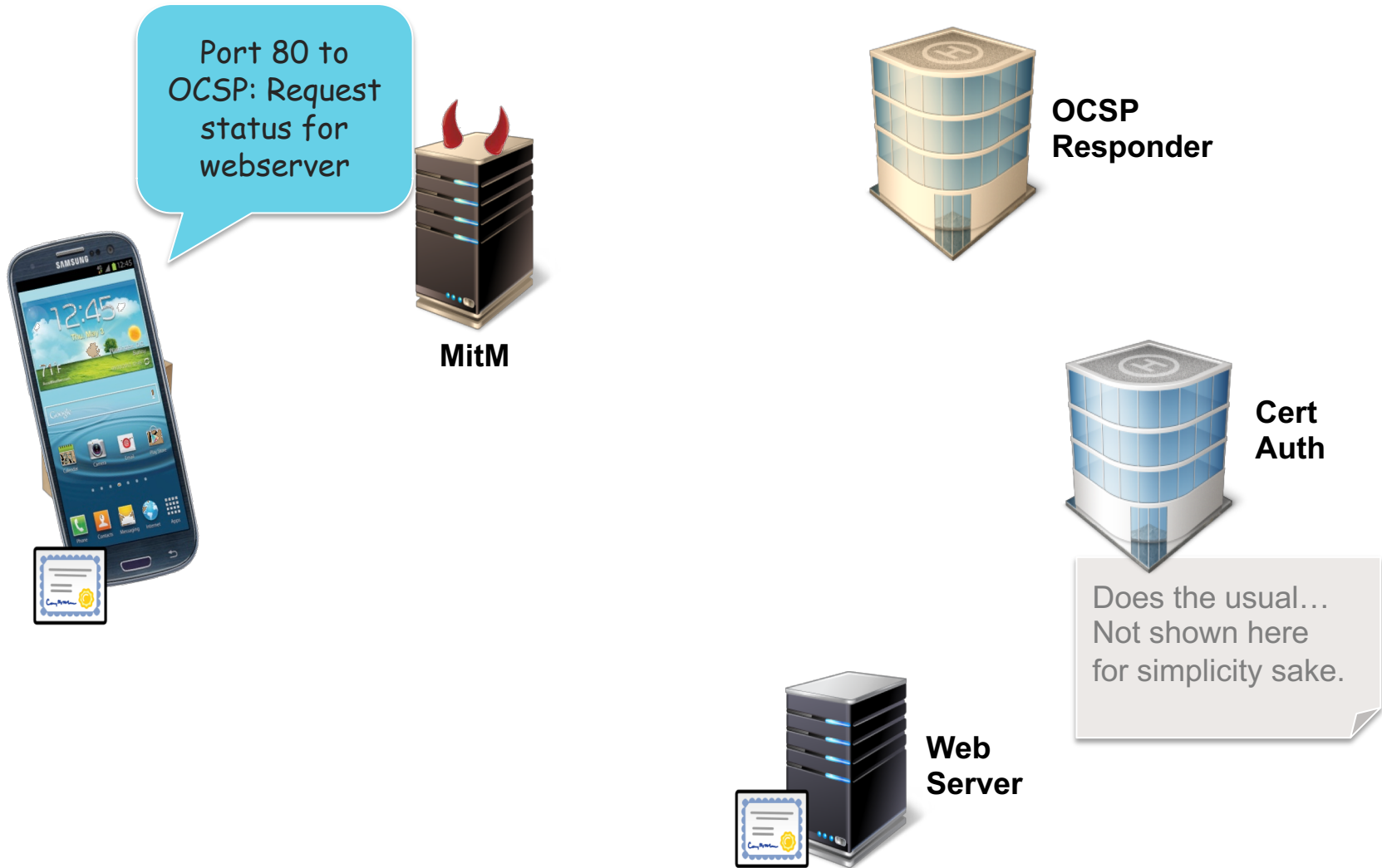
An alternative to certificate lists (CRL)



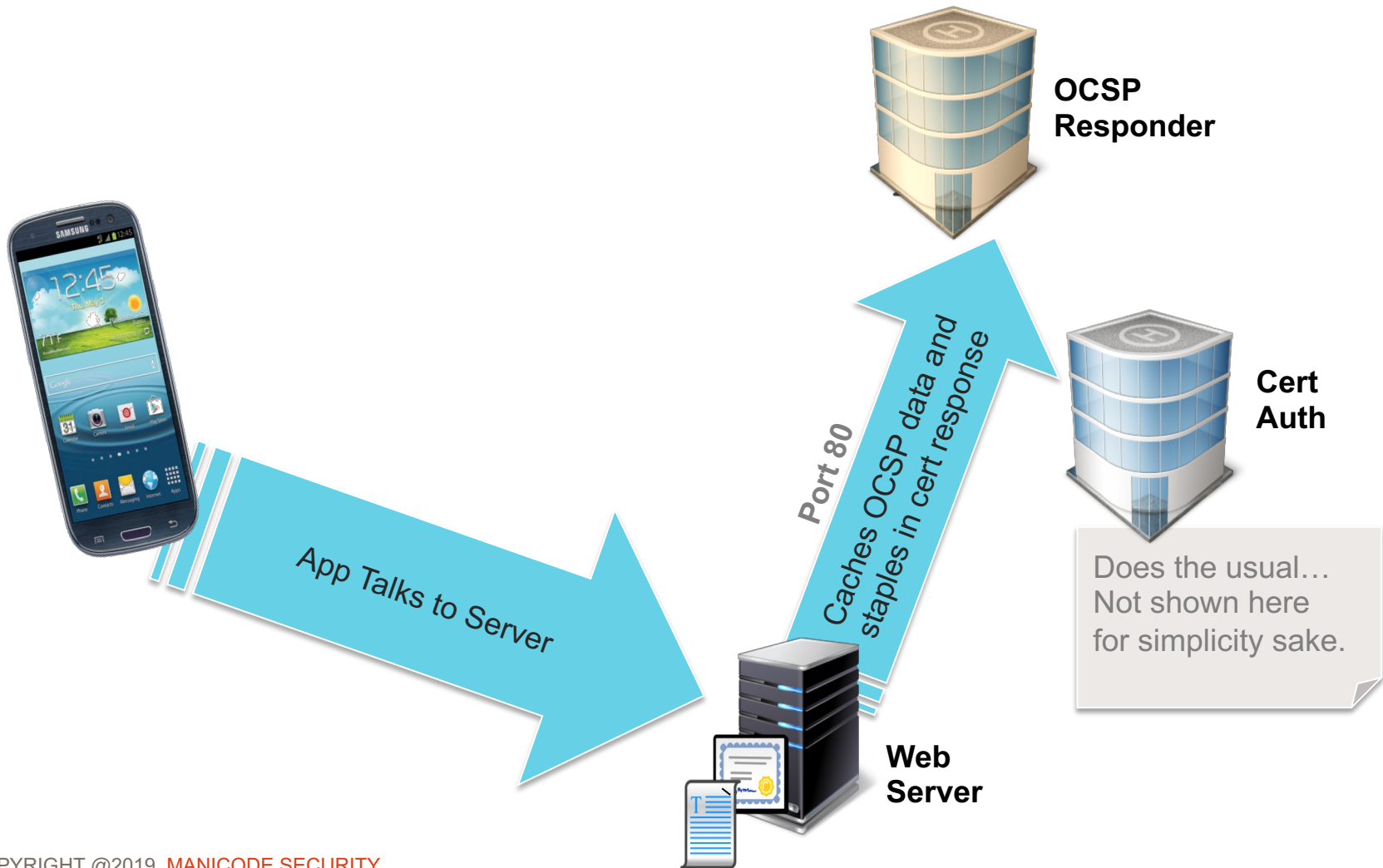
Attacks against OCSP



Attacks against OCSP



OCSP Stapling Faster, safer and more private




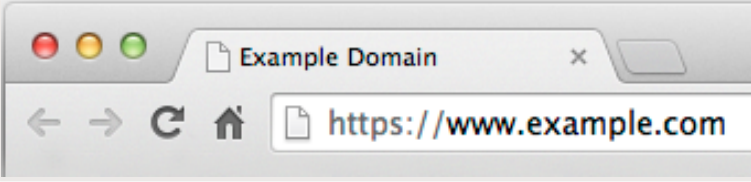
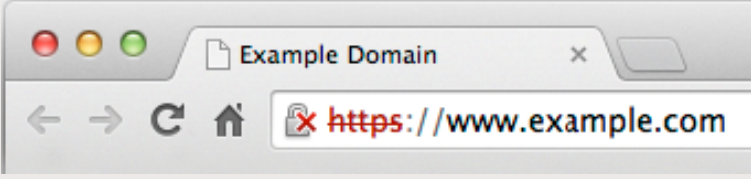
SHA-1 / RC4

SHA-1 and the Urgency to Move On

- **90% of websites** use SHA-1 to protect themselves from being impersonated
- In 2005, cryptographers proved that SHA-1 could be **cracked 2,000 times** faster than predicted
- **As long as browsers need to support SHA-1** for someone, anyone's **certificate can be forged** because browsers will not know there is a good cert that uses SHA-2

Year	Cost (In US\$)	Cost Within Reach For
2012	2,770,000	Government, large corporations
2017	700,000	Medium size institutions
2018	173,000	Organized crime
2021	43,000	University research

The Death of SHA-1 according to Google

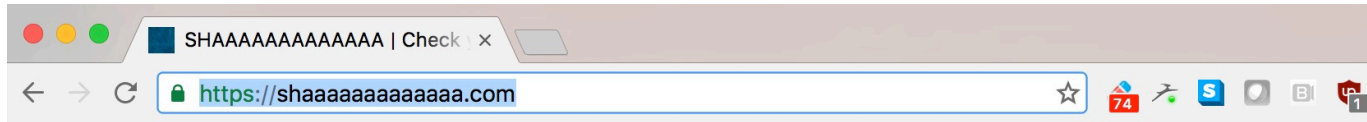
Chrome Version	Date	UI Changes	Behavior
39	Sept 2014		Certs that expire in Jan 2017 using SHA-1 or mixed content
40	Nov 2014		Certs that expire between 1 June 2017 to 31 Dec 2017 using SHA-1 in the chain
41	Q1 2017		Certs that expire on or after 1 Jan 2017

Source: <http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html>

The Death of SHA-1 according to Mozilla

- Show the “Untrusted Connection” error **whenever a publically issued SHA-1 certificate issued after January 1, 2016**, is encountered in Firefox.
 - Locally installed authorities (like MITM proxy tools) are NOT subject to this rule.
- Firefox will show the "Untrusted Connection" error message for all **SHA-1-based certificates after January 2017**.
- <https://www.fxsitecompat.com/en-CA/docs/2015/sha-1-based-certificates-with-validity-period-from-2016-will-not-be-validated/>

<https://shaaaaaaaaaaaaa.com/>



SHAAAAAAAAAAAAA

Check your site for weak SHA-1 certificates. [Open source](#), by [@konklone](#).

Check above to see if a site is still using certificates that were issued using the [dangerously outdated](#) SHA-1 signature algorithm.

As of **January 1, 2016**, no publicly trusted CA is allowed to issue a SHA-1 certificate. So any new certificate you get should automatically use a SHA-2 algorithm for its signature.

However, existing SHA-1 certificates are still trusted by modern browsers and operating systems. Generally, they will be removing support for SHA-1 entirely by January 1, 2017.

Legacy clients will continue to accept SHA-1 certificates, and it is possible to have requested a certificate on December 31, 2015 valid for 39 months. So, it is possible to see SHA-1 certificates in the wild that expire in 2019.

Credits

This website is an [open source](#) project, and includes a [command line tool](#) — please [lend a hand!](#)

Thanks to [Mathias Bynens](#), [Justin Mayer](#), and [Jonny Barnes](#) for inspiration and assistance.

SHA1 Collisions No Longer Theoretical



Secure

<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>

Google Security Blog

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

Announcing the first SHA1 collision

February 23, 2017

Posted by Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), Yarik Markov (Google), Alex Petit Bianco (Google), Clement Baisse (Google)

Cryptographic hash functions like SHA-1 are a cryptographer's swiss army knife. You'll find that hashes play a role in browser security, managing code repositories, or even just detecting duplicate files in storage. Hash functions compress large amounts of data into a small message digest. As a cryptographic requirement for wide-spread use, finding two messages that lead to the same digest should be computationally

Time to Retire RSA Key Exchange?

- Doesn't support forward secrecy
- Uses server key to encrypt the pre-master key created by the customer
- Weakness: Server key (part of the cert) is typically very long lived
- RSA (RC4 and DH) are not listed in Suite B, NSA and NIST approved list
- Keys grow significantly larger

NIST Recommended Key Sizes

Protection Target	Symmetric	DH or RSA	ECC
05 Years to protection against agencies	80	1024	160
20 Years to protection against agencies	112	2048	224
30 Years to protection against agencies	128	3072	256
Increase defense from quantum computers	256	15360	512

Conclusion

A Call to Action

1. **Make sure your ENTIRE WEBSITE or WEBSERVER is ALL HTTPS**
There is no excuse to not use HTTPS EVERYWHERE anymore
2. **HSTS**
Force the browser to always always use HTTPS and preload!
3. **Look into Certificate Transparency**
support as soon as possible
4. **Enable Certificate Authority Authorization** if possible
5. **Keep Updated**
Make sure to always use modern ciphers
6. **OCSP Stapling**
privacy, efficient and safer revocation

Listen to the madman...





It's been a pleasure.

jim@manicode.com

JIM MANICO

Secure Coding Instructor

www.manicode.com