PKI: Public Key Infrastructure

Certificates

- Cryptography provides confidentiality
- But: to whom am I talking?
- Someone who has the same key?
- With public key cryptography, everyone has their own
- But: how do I know to whom a given public key belongs?
- Answer: the *certificate*, a digitally-signed binding of a name to a public key

Certificates as a Systems Issue

- The basic concept is simple enough: someone digitally signs a record containing (at least) a name and a public key
- (Just as we signed a biometric template)
- But—it's more complicated than that

Public Key Infrastructure: PKI

- The Internet Security Glossary defines PKI as “The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.”
- Note carefully that the definition includes “people, policies, and procedures,” and not just code.
- N.B.: certificates were invented in 1978 in an MIT bachelor’s thesis

Issuing Certificates

- Typically, user generates a key pair, and presents the public key, identity, and proof of identity
- *Certificate Authority* (CA) signs the certificate and gives it back
- Note: certificates are self-secured; they can be verified offline

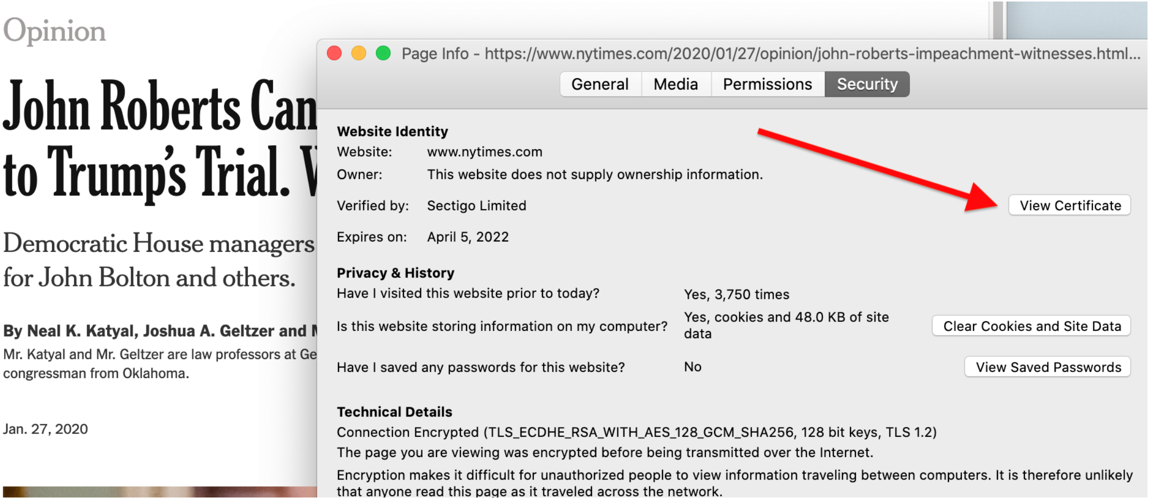
Certificate Authorities

- What makes a CA a CA?
- More precisely, how do you know who is authorized to issue a given certificate?
- And why do you trust them?

Trust Anchors

- *All* cryptographic trust has to start somewhere
- For any given application, you need to pick your trust anchors
- For the Web, that choice has been made for you: every OS or browser has a built-in CA
- Let’s look at the Columbia and Columbia CS web certificates

Checking a Certificate in Firefox



Opinion

John Roberts Can
to Trump's Trial. V

Democratic House managers
for John Bolton and others.

By Neal K. Katyal, Joshua A. Geltzer and M
Mr. Katyal and Mr. Geltzer are law professors at Ge
congressman from Oklahoma.

Jan. 27, 2020

Page Info – https://www.nytimes.com/2020/01/27/opinion/john-roberts-impeachment-witnesses.html...

Website Identity

Website: www.nytimes.com

Owner: This website does not supply ownership information.

Verified by: Sectigo Limited

Expires on: April 5, 2022

Privacy & History

Have I visited this website prior to today? Yes, 3,750 times

Is this website storing information on my computer? Yes, cookies and 48.0 KB of site data

Have I saved any passwords for this website? No

Technical Details

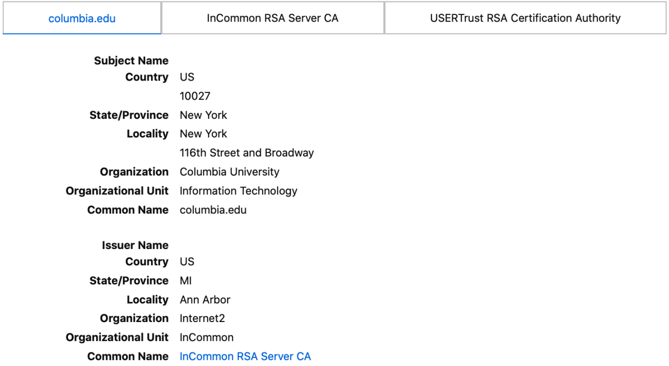
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

All desktop browsers have some way to display certificates

Columbia: The Basics



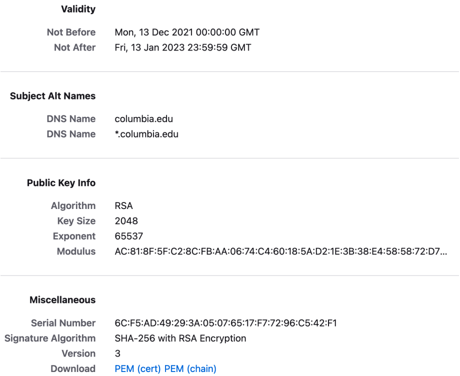
columbia.edu | InCommon RSA Server CA | USERTrust RSA Certification Authority

Subject Name	
Country	US
	10027
State/Province	New York
Locality	New York
	116th Street and Broadway
Organization	Columbia University
Organizational Unit	Information Technology
Common Name	columbia.edu
Issuer Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA

The certificate includes the organization name and address, and the CA's name and address.

Note the top: it shows the certificate chain from the trust anchor.

Columbia: Other Essential Data



Validity

Not Before	Mon, 13 Dec 2021 00:00:00 GMT
Not After	Fri, 13 Jan 2023 23:59:59 GMT

Subject Alt Names

DNS Name	columbia.edu
DNS Name	*.columbia.edu

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	AC:81:8F:5F:C2:8C:FB:AA:06:74:C4:60:18:5A:D2:1E:3B:38:E4:58:58:72:D7...

Miscellaneous

Serial Number	6C:F5:AD:49:29:3A:05:07:65:17:F7:72:96:C5:42:F1
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

Next: the validity period, actual domain names (called “Subject Alt Names”), and the key and algorithm.

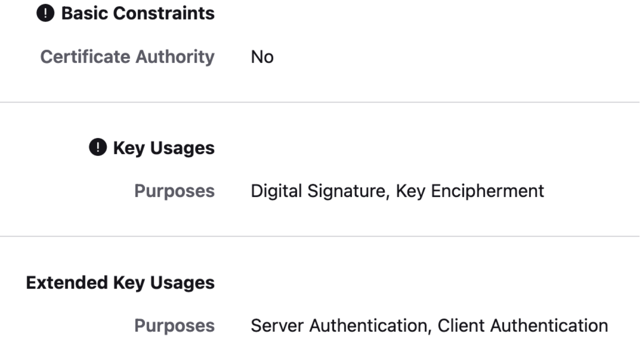
Domain Names

- The certificate holder’s identity follows a format originally intended for companies and their employees
- This certificate is for a web server—and browsers and users understand domain names, not “organizational units”
- Could a certificate authenticate part of a URL? In principle, yes, but I don’t think that the standards or the code support that

Expiration

- Why limit certificate lifetime?
- Limit the damage in case of private key compromise
- Limit the amount of revocation data that has to be kept
- Organizations have finite lifetimes—why leave their abilities around forever?
- And algorithms age

Columbia: Permissible Key Uses



❗ Basic Constraints

Certificate Authority	No

❗ Key Usages

Purposes	Digital Signature, Key Encipherment

Extended Key Usages

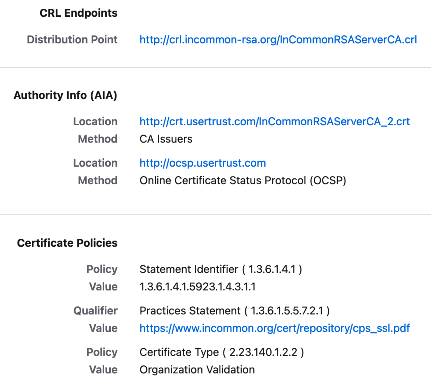
Purposes	Server Authentication, Client Authentication

The private key can be used for web sites, as a client and as a server, but cannot be used to issue new certificates or to sign code.

Why Limit Uses?

- Limit who can be a CA (but is that for protectionist reasons?)
- Limit how far you extend trust—that I can visit a website doesn't mean that I trust that website to run code on my computer
- Limit trust in employees who have access to the private key

Columbia: Policies and Revocation



CRL Endpoints

Distribution Point	http://crl.incommon-rsa.org/InCommonRSAServerCA.crl

Authority Info (AIA)

Location	http://crt.usertrust.com/InCommonRSAServerCA_2.crt
Method	CA Issuers
Location	http://ocsp.usertrust.com
Method	Online Certificate Status Protocol (OCSP)

Certificate Policies

Policy	Statement Identifier (1.3.6.1.4.1)
Value	1.3.6.1.4.1.5923.1.4.3.1.1
Qualifier	Practices Statement (1.3.6.1.5.5.7.2.1)
Value	https://www.incommon.org/cert/repository/cps_ssl.pdf
Policy	Certificate Type (2.23.140.1.2.2)
Value	Organization Validation

The certificate has the CA’s policies and revocation mechanisms.

How Do You Revoke a Certificate?

- Revocation is hard! Verification can be done offline; revocation requires some form of connectivity
- Publish the URL of a list of revoked certificates in a Certificate Revocation List (CRL)

  ☞ One reason for certificate expiration dates; you don't need to keep revocation data forever
- Online status checking (OCSP)
- Note: OCSP has privacy implications
- STU-IIIs use flooding algorithm—works well because of comparatively closed communities

Why Revoke Certificates?

- Private key compromised
- Cancel authorization associated with certificate
- CA key compromised, e.g., DigiNotar
- Algorithm compromised (rare use)

Verifying Identity

- A CA verifies some site's identity. How?
- It's easy to verify the existence of a corporation—but how do you verify who speaks for it?
- For publicly traded companies, top executives are generally a matter of public record—do the CEO or CTO have to show up in person?

Many Different Ways!

From the Baseline Requirements document from the CA Browser Forum, verification can be done by:

- Documents from or communication with a government agency;
- An in-person visit by the CA;
- A reliable third-party database;
- An “attestation letter” from a lawyer or accountant, etc.;
- More. . .

These procedures are manual, annoying, and complex—can we automate them?

Automated Web CAs

- Web site certificates should be issued to the owner of the web site, e.g., the party that controls the site
- Demonstrate control of the web site by doing something only the site owner can do
- Example: the CA sends the site a random number; the site puts that number into a specific URL

Let’s Encrypt

- The Electronic Frontier Foundation started an automated CA, letsencrypt.org, to issue certificates based on that principle
- There are open source packages to request and install such certificates
- There is no equivalent for code-signing certificates—why not?
- How would you attack this scheme?

Attacking Letsencrypt.org

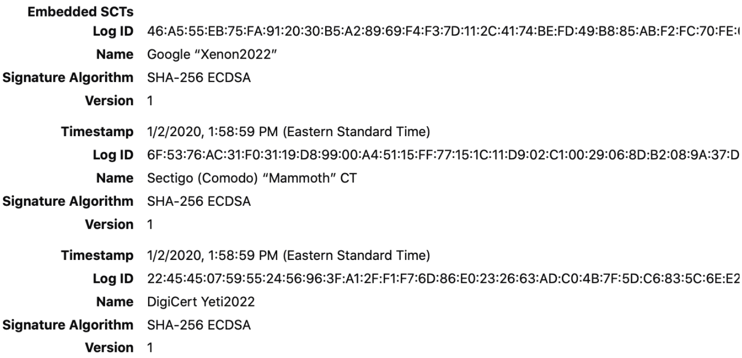
Attacking Letsencrypt.org

- The verification step depends on reaching the correct web site
- That means that any attack that sends someone to the wrong web site can result in a bogus certificate being issued
- Example: playing games with BGP routing, or just hacking the site

Using a Bogus Certificate

- What can an attacker do with a bogus certificate for a site?
- It permits impersonation of the site—but that requires that that the attacker have the ability to redirect or intercept user traffic
- That’s easy under certain assumptions: you’re an ISP, a government, on the same LAN as the victim, etc.
- It probably does not permit reading old traffic—modern versions of TLS use *forward secrecy* precisely to prevent that

Columbia: Certificate Transparency



```
     Embedded SCTs
            Log ID  46:A5:55:EB:75:FA:91:20:30:B5:A2:89:69:F4:F3:7D:11:2C:41:74:BE:FD:49:B8:85:AB:F2:FC:70:FE:
              Name  Google “Xenon2022”
Signature Algorithm SHA-256 ECDSA
           Version  1

         Timestamp  1/2/2020, 1:58:59 PM (Eastern Standard Time)
            Log ID  6F:53:76:AC:31:F0:31:19:D8:99:00:A4:51:15:FF:77:15:1C:11:D9:02:C1:00:29:06:8D:B2:08:9A:37:D
              Name  Sectigo (Comodo) “Mammoth” CT
Signature Algorithm SHA-256 ECDSA
           Version  1

         Timestamp  1/2/2020, 1:58:59 PM (Eastern Standard Time)
            Log ID  22:45:45:07:59:55:24:56:96:3F:A1:2F:F1:F7:6D:86:E0:23:26:63:AD:C0:4B:7F:5D:C6:83:5C:6E:E2
              Name  DigiCert Yeti2022
Signature Algorithm SHA-256 ECDSA
           Version  1
```

Last: the certificate transparency servers that list the issuance.

Web Trust Anchors

- *All* browsers trust many CAs
- Last time I counted, Firefox trusts about 200 CAs; Windows IE trusts > 300—and at least 10% are agencies of some government

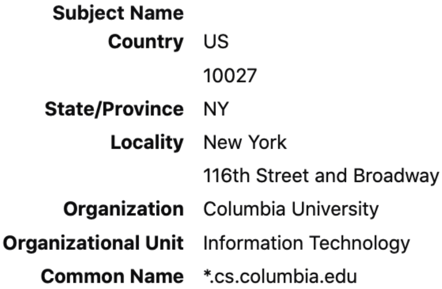
☞ All browser vendors have their own policies for adding CAs, and there are industry standards

- *Any* trusted CA can issue a certificate for *any* domain
- When DigiNotar was hacked, bogus certificates were issued for gmail.com, yahoo.com, mozilla.com, and many more
- How do we prevent this?

Certificate Transparency

- Concept: all issued certificates are logged in one or more transparency servers
- Legitimate certificate holders can query these servers to see if there are any bogus certificates listing them
- Idea originally pushed by Google.
- (I was skeptical that it would catch on, but Google has a lot of influence. . .)

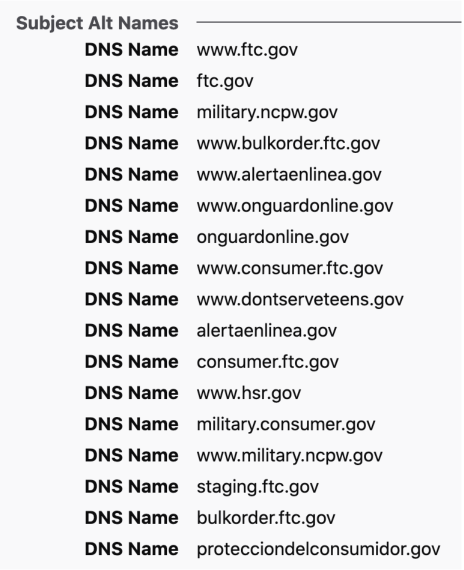
The CS Department Certificate



Subject Name	
Country	US
	10027
State/Province	NY
Locality	New York
	116th Street and Broadway
Organization	Columbia University
Organizational Unit	Information Technology
Common Name	*.cs.columbia.edu

Note that it was CUIT that got the CS department’s certificate—but CUIT did not issue it itself. Why not?
On the one hand, CUIT knows CU CS and who here is authorized to have the certificate. On the other, CUIT is not a CA—and that may be wise.

The Federal Trade Commission’s Certificate



Subject Alt Names

DNS Name	www.ftc.gov
DNS Name	ftc.gov
DNS Name	military.ncpw.gov
DNS Name	www.bulkorder.ftc.gov
DNS Name	www.alertaenlinea.gov
DNS Name	www.onguardonline.gov
DNS Name	onguardonline.gov
DNS Name	www.consumer.ftc.gov
DNS Name	www.dontserveteens.gov
DNS Name	alertaenlinea.gov
DNS Name	consumer.ftc.gov
DNS Name	www.hsr.gov
DNS Name	military.consumer.gov
DNS Name	www.military.ncpw.gov
DNS Name	staging.ftc.gov
DNS Name	bulkorder.ftc.gov
DNS Name	protecciondelconsumidor.gov

- Look at all of the domain names listed
- How does the CA—a commercial entity, not part of the government—know which `.gov` domains the FTC actually runs?
- (There are many more than I’ve shown)

Certificate Practices Statements

- All CAs publish Certificate Practices Statements (CPS)
- Have you ever read one?
- Let's look at the CPS for Internet2 InCommon, which issued CU's certificate (https://www.incommon.org/custom/certificates/repository/cps_ssl.pdf)

Liability

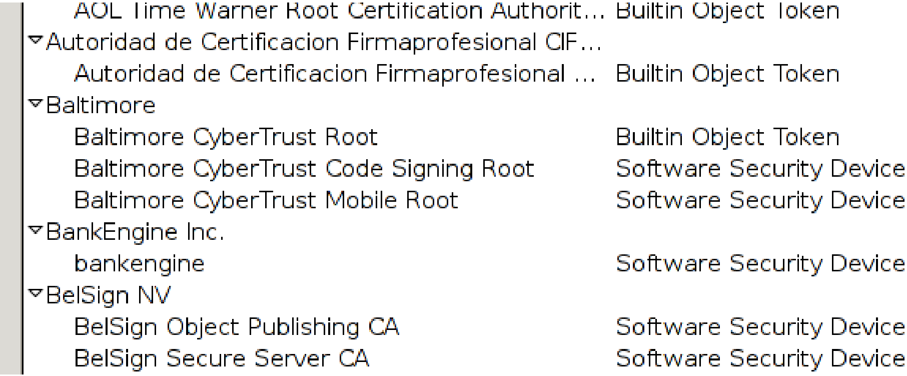
- CAs are inherently about trust
- But: “InCommon provides no warranty as further described in the Subscriber Addendum and Relying Party Agreement.”
- (Lots of capital letters emphasizing that there is no warranty. . .)
- “Relying Party Agreement”?

Relying Party Agreement

- “**Relying Party** An entity that relies upon the information contained within the Certificate.
- “**Relying Party Agreement** An agreement that must be read and accepted by a Relying Party prior to validating, relying on or using a Certificate and is available for reference in the InCommon PKI repository as described in Section 2.
- “Relying parties use InCommon’s PKI service certificates to perform transactions, communications, or other functions at their own discretion.”
- I’m sure you’ve all read and accepted this document you’ve never heard of before visiting, say, the CS department web site. . .
- Oh yes: they give the URL for the repository with the Relying Party Agreement, but it’s not a hyperlink

CAs as Corporations

- A CA is typically a corporation or a unit of a corporation; as such, it is subject to the usual corporate life
- Example: Baltimore CyberTrust



AOL Time Warner Root Certification Authorit…	Builtin Object Token
▿ Autoridad de Certificacion Firmaprofesional CIF…	
Autoridad de Certificacion Firmaprofesional …	Builtin Object Token
▿ Baltimore	
Baltimore CyberTrust Root	Builtin Object Token
Baltimore CyberTrust Code Signing Root	Software Security Device
Baltimore CyberTrust Mobile Root	Software Security Device
▿ BankEngine Inc.	
bankengine	Software Security Device
▿ BelSign NV	
BelSign Object Publishing CA	Software Security Device
BelSign Secure Server CA	Software Security Device

It's amusing to read Baltimore CyberTrust's complex corporate history

Baltimore CyberTrust

- CyberTrust was originally a subsidiary of GTE
- Sold to Baltimore Technologies, an Irish company, in 2000
- In 2003, they sold CyberTrust to BeTrusted Holdings
- In 2004, BeTrusted merged with TruSecure Corporation
- In 2007, acquired by Verizon—which itself included the old GTE
- In 2015, sold to DigiCert
- Query: which management structure did someone decide to trust?

☞ Matt Blaze: “a commercial CA will protect you from anyone from whom they wouldn’t take money”

Types of Certificates

- Web certificates: used for TLS connections
- Email certificates—sign and encrypt email
- IP address range certificates—used to secure routing announcements
- Code-signing certificates—authenticate the source of a program
- ☞ Used for market control, i.e., Apple's App Store, and as a security measure
- Many more types

Rights Granted by Certificates

- Typically, none
- *Most* certificates are assertions of identity
- It’s still up to the relying party to decide what authorizations to grant that identity

Authorization Certificates

- Identity-based: some organization, such as Verisign, vouches for your identity
  - ☞ Cert issuer is not affiliated with verifier
- Authorization-based: accepting site issues its own certificates
  - ☞ Cert issuer acts on behalf of verifier
- Identity-based certificates are better when user has no prior relationship to verifier, such as secure Web sites
- Authorization-based certs are better when verifier wishes to control access to its own resources—no need to trust external party
- I call this “pki” instead of “PKI”

Why Aren't There More Authorization Certificates?

- Certificates have been granted a certain mystique
- They're somehow seen as different from a syadmin adding a username and password to a system
- From the InCommon CPS: "For InCommon CA key recovery purposes, the InCommon CA signing keys will be encrypted and stored within a secure environment. InCommon's escrowed copy of the private key is split across a number of removable media and requires a subset of the number to reconstruct. Custodians in the form of two or more InCommon authorized officers are required to physically retrieve the removable media from the distributed physically secure locations."

Mystique

“Organizations are regularly told that they are complex, require ultra-high security, and perhaps are best outsourced to competent parties. Setting up a certificate authority (CA) requires a “ceremony”, a term with a technical meaning but nevertheless redolent of high priests in robes, acolytes with censers, and more. This may or may not be true in general; for most IPsec uses, however, little of this is accurate. (High priests and censers are definitely not needed; we are uncertain about the need for acolytes. . .)”

Can You Run Your Own CA?

- Yes, but it’s hard
- It’s not *inherently* hard, but the documentation is pretty poor
- There are decent web pages on how to create “self-signed certificates”
- Should I give a homework assignment on creating certificates? Maybe. . .

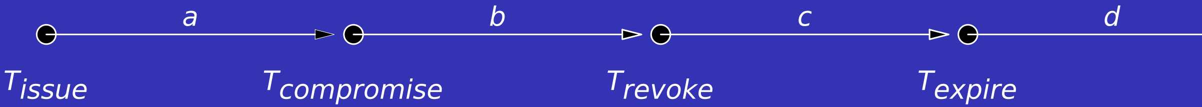
Certificate Lifecycle

1. Generate a key pair
2. Request a certificate from a CA
3. Use the certificate
4. The private key is compromised
5. Revoke the certificate

Doing Revocation in Practice

- What should be done if a CRL can't be retrieved? What if retrieval failures is due to a DDoS attack on the CRL site?
- OCSP seems better—less time between compromise and notification
- Slow—it requires another TLS connection from the browser (which is several round trips)
- And what if the OCSP server is unreachable or returns "Unknown"?
- Besides, the time advantage may be more imaginary than real

Certificate Compromise Timing



a b c d

T_{issue} $T_{compromise}$ T_{revoke} T_{expire}

- OCSP shrinks the tiem between compromise and revocation
- But what is $T_{discovery}$?
- Almost always, it is much larger than b, even for CRLs
- That is, key compromises are rarely discovered promptly.
- You are insecure during the period between compromise and revocation, not the period between discovery and revocation
- OCSP helps, but not by nearly as much as one would think

Using “Dead” Certificates

- How should one treat expired or revoked certificates?
- Clearly, one should not set up new TLS connections
- But what about a digitally signed document or file?
- Was it signed before or after revocation?
- How can you tell? (Did an attacker reset the system clock to sign a malicious file?)
- In principle, trust signatures created before compromise or at least before detection, but it’s hard for outsiders to know
- (Btw: the timestamp in a revocation notice is ambiguous: it may be the time the compromise was detected, or perhaps when it was reported. Check the CPS.)

Web of Trust

- PGP use a “web of trust” — rather than a tree, certificates form an arbitrary graph
- *Anyone* can sign a certificate
- Most people have more than one signature — I have more than signatures on my primary PGP key
- Do you know and trust any of my signers?
- See my key at `http://www.cs.columbia.edu/~smb/smbpgp.txt`

Does the Web of Trust Work?

- Number of signatures alone is meaningless; I can create lots of identities if I want
- I can even forge names — is the “Luca Carloni” who signed my key the same one who’s a professor here? How do you know?
- There are at least six PGP keys purporting to belong to “George W. Bush”. One is signed by “Yes, it’s really Bush!”
- You have to define your own set of trust anchors, as well as policies on how long a signature chain is too long

Security Analysis: Is PKI Secure?

- What are the weak points?
- Who can attack them?

Attacking the Cryptography

- Today’s standards—SHA256; 2048-bit RSA or 256-bit ECDSA—are believed to be secure
- But when web encryption started, people used MD5 and 1024-bit RSA; both are currently believed to be insecure
- Some sites continued using older algorithms well after they were known to be bad
- And there have been exploits—but probably by intelligence agencies

Flame

- The Flame malware relied on a software vendor using MD5 in its certificates, well after they should have stopped (and well after Windows should have rejected such certificates)
- MD5 was known to be weak—but Flame used a previously unknown cryptanalytic mechanism to attack MD5
- Most hackers are not expert cryptanalysts. . .
- Flame also exploited a design error by Microsoft and hijacked the Windows Update mechanism

Microsoft and MD5

- Microsoft has excellent cryptographers; they *knew* that MD5 was broken
- But: they had to preserve backwards compatability for long enough for all of their customers to migrate to a stronger hash function
- Sites that had upgraded their own infrastructure could still be vulnerable to Microsoft's decision

Private Key Compromise

- Sites’ private keys can be stolen
- Stuxnet relied on two different stolen vendor code-signing keys
- (Were these keys in HSMs? They should be)

Identity Spoofing

- In 2001, someone impersonated Microsoft and got two fake code-signing certificates from VeriSign
- VeriSign employees did not check the requester’s credentials properly
- But: their back-end audit process caught the problem
- CRLs weren’t really used then, so Microsoft shipped a patch to ignore those two certificates

Buggy Code

- Windows 10 had a certificate-checking bug: under some circumstances, it would accept *anything*
- This bug was found by the NSA, which thought it was so scary it told Microsoft and issued its own advisory

Blind Trust

- Code-signing certificates are (at best) statements of where code came from, not that it's harmless
- Some hackers have purchased genuine code-signing certificates from Apple or Microsoft
- Victims' operating systems accept the malware, because it's digitally signed
- There is a confusion of identity with authorization and/or benignity
- These were identity certificates that were treated as authorization certificates!

Hacking

- Hackers, believed to be linked to Iran, compromised the DigiNotar and Comodo CAs
- The DigiNotar private keys were stored in an HSM, but the attacker controlled the computer that controlled the HSM, and hence was able to get it to sign bogus certificates
- The attack put DigiNotar out of business

Conclusions

- Certificates are useful but they aren't panaceas
- Buggy code can beat good crypto
- Note the analysis: at some point, *every piece* of the security chain for certificates has been compromised

Questions?

(Peregrine falcon, Central Park, January 26, 2022)