ELEC5616 COMPUTER & NETWORK SECURITY

Lecture 16:
Overview of SSL/TLS

RAW HTTP IS INSECURE

In stock standard HTTP, everything goes across the line in plaintext

This leaves connections vulnerable to

– message tampering

– eavesdropping on connections

– man-in-the-middle attacks

On top of that, there are numerous flaws in TCP/IP that we'll discover in the following weeks

INTRODUCING SSL/TLS

Transport Layer Security (TLS) with its predecessor Secure Socket Layer (SSL) are cryptographic protocols for secure communication

– Asymmetric cryptography for authentication
– Symmetric encryption for confidentiality of messages
– MACs for integrity

HTTP sits on top of that to produce HTTPS (HTTP Secure):
the only information that leaks is the IP and port you're connecting to

Previously only considered to be used for "secure operations"
Now it's suggested it's used everywhere by default
(12% of all websites now use HTTPS)

TLS PROTOCOL

Client	=>	SSL version, available ciphers, other info	=>	Server
Server	=>	SSL v, select cipher, send certificate	=>	Client

Client authenticates the server (messages sent should be signed by cert)

Client (possibly in conjunction to server, depending on cipher) create the pre-master secret for the session, encrypts using server's cert, sends to the server

(Optional) Server may authenticate the client using client's certificate

Client	=>	All messages from here will be encrypted	=>	Server
Server	=>	All messages from here will be encrypted	=>	Client

WHERE DO THESE CERTIFICATES COME FROM?

In SSL/TLS, the server sends across a certificate.

How can we actually decide to trust this?

Certificate Authority (CA): trusted third party that issues digital certificates.

Who are these Certificate Authorities?

Mozilla ships with a list of 36 trusted root CA
Small number of multinational companies – significant barrier to entry

Symantec (owns *VeriSign*, *Thawte* and *Geotrust*)	42.9%
Comodo	26%
GoDaddy	14%
GlobalSign	7.7%

ISSUES WITH CERTIFICATE AUTHORITIES

Do we actually trust these trusted third parties?

– Many have horrific security practices or willing to work with governments

– The small number of Root CAs allow other CAs to sign on their behalf

– In 2011, fraudulent certificates obtained from Comodo used for man-in-the-middle in Iran – they can man-in-the-middle *any website* they decide

Sell based on the most ridiculous things

– Cost of 128 bit and 256 bit SSL certificates are commonly different even though next to no more work goes in to creating it

Saying “this website secured by SSL” gives users a false sense of confidence

– It’s beyond trivial for scammers to get an SSL certificate...

HOW DO YOU ACQUIRE ONE?

Domain Validation

Send an email to domain to email in WHOIS, postmaster@, root@, etc

(faulty theory: only the owner of the domain can access those mailboxes)

Extended Validation Certificates

CA verify the proposed owner passes a set of identity verification criteria

Exactly the same as normals certs except (a) cost more and (b) green bar

(doesn’t prevent a faked normal cert from intercepting traffic)

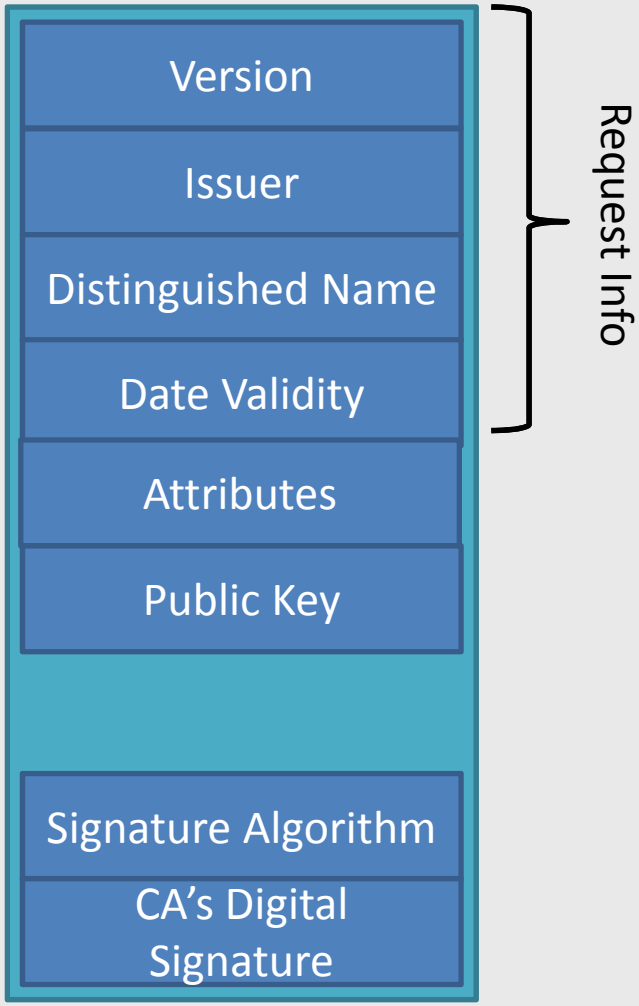
HOW DOES IT WORK?

- Generate a public-private key pair for your server
- Generate a Certificate Signing Request (CSR)

CSR contains any relevant details + your public key

[note: distinguished name => *example.com*]

- Send CSR to the CA
- CA confirms any necessary details (primarily domain verification)
- The CA signs your certificate, allowing users to trust the certificate
- Install your certificate on your web server



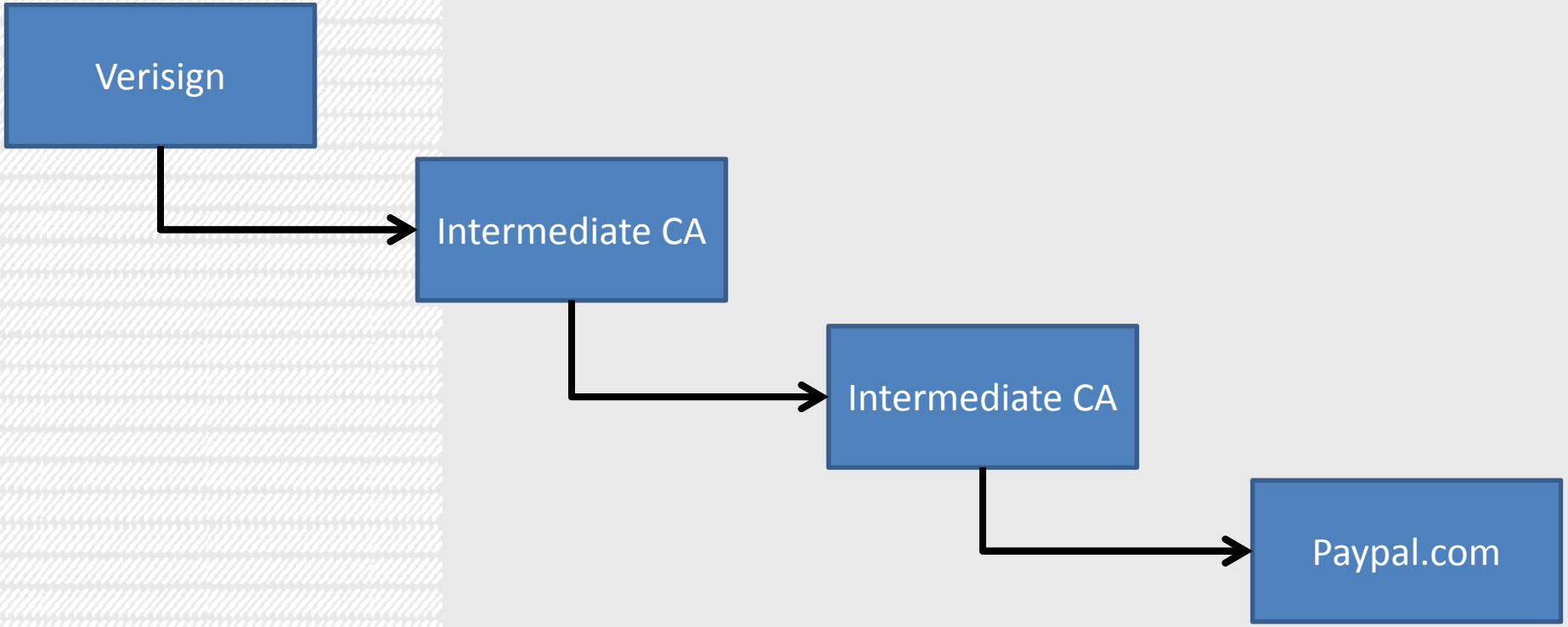
Now when a user connects to your server, you send your signed certificate

Cert provides the public key and provides the signature by CA

User checks their local CA cert verifies the given server's signature

CERTIFICATE CHAINING

- Root CAs can give permission to other CAs to sign SSL certificates



CERTIFICATE REVOCATION

There is a method to revoke SSL certificates called:

Online Certificate Status Protocol (OCSP)

Whenever a browser sees a new SSL cert, it makes a request to the OCSP URL that's embedded in the CA's signing certificate

The OCSP URL sends a signed response indicating if the certificate is still valid

The signature only covers some of the response data and specifically doesn't cover the response status. Thus, we can send back any response status.

Thus, attackers can intercept OCSP requests and reply with a "*tryLater*" response status. All browsers think this is fine and avoid raising an issue.

EVEN IF SSL WERE PERFECT...

There are many attacks on SSL even if it were perfect.

Many of these take advantage of users and the inability for browsers to correctly recommend courses of action when things “aren’t right”.

(see *SSLSniff* in following lectures)

BEAST AND RC4

BEAST (Browser Exploit Against SSL/TLS) is a practical exploit taking advantage of a protocol flaw with Cypher Block Chaining (CBC) in TLS 1.0

Implication: if an attacker can read and inject packets quickly, they can eavesdrop on any SSL connection using *-CBC (i.e. AES-CBC)

(useful in local networks: WIFI in coffee shops, LAN in labs, etc)

TLS 1.1 (released 2006) fixed this, but TLS 1.1 still not widely adopted in 2012!

Security Experts: Avoid AES-CBC, use RC4!

(remember: RC4 is a stream cipher and doesn't need CBC)

BEAST AND RC4

RC4 is faster and cheaper than AES-CBC – heavily used by Google due to this

Minor issues: some government standards *require* AES-CBC, disallow RC4

Major issue: RC4 has shown small but significant bias (preference toward a one or zero depending on input) in the first 256 bytes of ciphertext output

You need many RC4 ciphertexts (2^{32} gives guaranteed plaintext recovery) to exploit this if it's *random*, but this implies RC4 has severe exploitable flaws.

Worse: what's at the first few bytes of most SSL/TLS connections? *The cookie*. If you can steal the cookie, you can steal the identity of the user.

*So now RC4 is vulnerable and *-CBC as well (until TLS 1.1 is widely supported)*

REQUIRED VIEWING

DEFCON Presentation:

"*More Tricks for Defeating SSL In Practice*"

Moxie Marlinspike