Should You Trust the Padlock?
Web Security and the “HTTPS Value Chain”

Keeping Current
20 November 2013

Ken Calvert
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Threats:
What are we afraid of?

**Eavesdropping**: sensitive information carried in HTTP messages can be read by intruders.
Threats:
What are we afraid of?

Impersonation: Bob thinks he is talking to Amazon, but it’s really Trudy’s fake site.
Impersonation: Bob thinks he is talking to Amazon, but it’s really Trudy’s fake site.

Trudy can exploit Bob’s (misplaced) trust to compromise his system!
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Countermeasures

Confidentiality: end-to-end encryption prevents eavesdropping

amazon.com
Countermeasures

**Confidentiality:** end-to-end encryption prevents eavesdropping
Authentication: Bob can tell if he is talking to the real Amazon.com. (More precisely: his browser can.)
Securing the Web

• Secure Sockets Layer (Netscape) and Transport Layer Security (IETF) were developed (ca. 1995-6) to secure the channel between client and server
  – Confidentiality: Prevent eavesdropping
  – Authentication: Detect impersonation

• These are general protocols, designed for use by any application running over TCP
  – HTTPS = Hypertext Transfer Protocol over SSL/TLS
    Both SSLv3 and TLSv1-3 are in common use, but only TLS is still being updated with new ciphersuites

• Both use public key cryptography to authenticate the server and establish confidentiality

• Authentication turns out to be the main challenge
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Public Key Cryptography

• Basic idea:
  – Keys come in pairs
    • one key is public (known to anyone)
    • one key is private (known only to Bob)
  – Basic operations:
    signature = \text{sign}(\text{message, private key})
    \text{verify}(\text{signature, public key}) \rightarrow \text{valid | invalid}

• Mathematics of the algorithm (plus assumptions about hardness of certain problems) ensures that a valid signature cannot be created without knowledge of the private key.
Public Key Authentication

Authentication: Bob can tell if he is talking to the real Amazon.com. (More precisely: his browser can.)

```
sig = sign(string, key)
```

“please prove you are amazon.com by signing this random bit string: 0xf3a71b0948862e11a798015b2c20”

signature = 0x371f1c...

amazon.com’s private key

amazon.com’s public key
Public Key Authentication

**Authentication**: Bob can tell if he is talking to the real Amazon.com. (More precisely: his browser can.)

```
verify(signature, pub key) = valid 😃
```

**Key Question**: How does Bob learn Amazon.com’s public key?
Public Key Distribution

Ways for Bob to learn Amazon.com’s public key:
- Ask the server?
  - No: This is begging the question!
    We don’t know we’re really talking to Amazon!
Public Key Distribution

Why Bob can’t just ask the server for its public key...

“Hey Amazon, please tell me your public key.”

“Why sure, sonny, it’s 0x119b3e...”
Public Key Distribution

Ways for Bob to learn Amazon.com’s public key:

• Ask the server?
  – No: This is begging the question!
    We don’t know we’re really talking to Amazon!

• The key comes pre-installed in the browser?
  – No: This doesn’t scale!
    Millions of sites need HTTPS; new ones may arise every day.
Public Key Distribution

Ways for Bob to learn Amazon.com’s public key:

• Ask the server?
  – **No**: This is begging the question!
    We don’t know we’re really talking to Amazon!

• The key comes pre-installed in the browser?
  – **No**: This doesn’t scale!
    *Millions* of sites need HTTPS; new ones may arise every day.

• **Trusted 3rd parties** certify the binding between entity and public key (by **signing** the binding)
  – Browser comes equipped with the public keys of a **limited number** of these **Certificate Authorities**
A trusted 3rd party attests to Amazon’s public key

Reduces the problem:
1. Get the CA’s public key.
2. Given a server’s (amazon.com’s) certificate (issued by that CA), verify the CA’s signature on the cert.
3. Use the certified public key to verify the server’s identity

CA public key is a root of trust
- CA can sign keys of other CAs and/or end users (amazon)
- Scales (as usual) by adding hierarchy
Certificate Authorities (CAs)

• CA Public keys are distributed out-of-band
  – Usually in the form of a self-signed certificate
  – Browsers come preconfigured with CA certs

• In general, the job of a CA is to make sure that it only issues certificates that are legitimate.
  – What should you have to do to get a certificate?
    • Tradeoff: ease of acquiring vs. ease of impersonation
CA Public Keys in Browsers

When a server requests my personal certificate:
- Select one automatically
- Ask me every time

Options:
- View Certificates
- Validation
- Security Devices
CA Public Keys in Browsers

### Certificate Name | Security Device
---|---
(c) 2005 TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş., TÜRKTRUST Elektronik Sertifika Hizmet Sağlayıcısı | Built-in Object Token
128.163.141.21 | Software Security Device
128.163.141.21 | Built-in Object Token
AC Camerfirma S.A. | Built-in Object Token
Chambers of Commerce Root - 2008 | Built-in Object Token
Global Chambesign Root - 2008 | Built-in Object Token
AC Camerfirma S.A. Cif A827643287 | Built-in Object Token
Chambers of Commerce Root | Built-in Object Token
Global Chambesign Root | Built-in Object Token
Actalis S.p.A./03358520967 | Built-in Object Token
Actalis Authentication Root CA | Built-in Object Token
AddTrust AB | Built-in Object Token
AddTrust Public CA Root | Built-in Object Token
AddTrust Qualified CA Root | Built-in Object Token
AddTrust Class 1 CA Root | Built-in Object Token
AddTrust External CA Root | Built-in Object Token
COMODO Certification Authority | Software Security Device
COMODO SSL CA 2 | Software Security Device
COMODO High-Assurance Secure Server CA | Software Security Device
COMODO SSL CA | Software Security Device
PositiveSSL CA 2 | Software Security Device
InCommon Server CA | Software Security Device
Network Solutions Certificate Authority | Software Security Device
"Common Name" = DNS name of the server that uses HTTPS
If Certificate Validation Fails...

This Connection is Untrusted

You have asked Firefox to connect securely to fastlane.nsf.gov, 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

fastlane.nsf.gov uses an invalid security certificate.

The certificate is only valid for www.fastlane.nsf.gov

(Error code: ssl_error_bad_cert_domain)

I Understand the Risks
Certificate Validation Fails...what to do?

This Connection is Untrusted

You have asked Firefox to connect securely to fastlane.nsf.gov, 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

fastlane.nsf.gov uses an invalid security certificate.

The certificate is only valid for www.fastlane.nsf.gov

(I Understand the Risks)
CA Public Keys in Browsers

• Firefox comes with 130+ roots of trust (CAs public keys) pre-installed
  – Other browsers similar, but...
• Roots of trust may vary with browser and platform
Trust Structures

Basic Question: what authorities do I trust?

• Monopoly
  – Single root of trust, everybody knows its key, which never changes
  – Obvious problems

• Hierarchy of CAs
  – Root certifies “child” CAs, which may certify other CAs or regular users
  – Benefit: easier to get to a CA near you
  – Drawback: still a single root of trust
Trust Structures

• Web of Trust
  – Individuals (Alice, Bob) sign keys of people they trust
  – I collect public keys of people I know
  – When presented with a new public key, try to find a chain of people I trust, ending with someone who signed it
  – This is used in Zimmerman’s PGP (“Pretty Good Privacy”)
  – Issue: scalability, reliability

• “Oligarchy” (name due to Kaufman, Perlman and Speciner)
  – Multiple roots of trust, each signs certificates
  – Trust only public keys signed by one of these CAs
  – How to choose a CA?
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Levels of Certification

Certificates come in different "levels":

- **Domain Validated (DV)**
  - Issuing CA verifies "control" of the domain name
    - In practice: answer an email to the address listed in the SOA record of the DNS zone (WHOIS database)
    - Process can be automated ⇒ fast turnaround

- **Organization Validated (OV)**
  - No standards for what this means
  - Typical: verify organization’s contact information via third party source (Secretary of State, telephone directory, ...)

- **Extended Validation (EV)**
  - More extensive validation process
  - More expensive
  - Browser indication: "green bar"
Who are the Stakeholders?

- Website Owners
- Certificate Authorities
- Relying Parties (Users)
- Browser “Vendors”
Who Risks What?

Risk: **loss of business** if users don’t believe their site is secure

Risk: **loss of sensitive information, time, $$** if they trust a rogue site

- **Website Owners**
- **Certificate Authorities**
- **Relying Parties (Users)**
- **Browser “Vendors”**
Who Risks What?

- **Risk:** *loss of market share* if browser trusts a bad CA
- **Risk:** *bankruptcy* if removed from browser trust chain

- **Browser “Vendors”**
- **Website Owners**
- **Certificate Authorities**

**Relying Parties (Users)**
What Does the Market Look Like? [1]

• EFF’s SSL Observatory project (December 2010)
  – Collected 1.5M valid certificates from around the web
  – Identified ~1100 issuing CAs

• Highly concentrated
  – 3 vendors account for more than ¾ of the market
    • Symantec (includes Verisign and Thawte)
    • GoDaddy
    • Comodo

• Widely varying prices

# Price Variations

<table>
<thead>
<tr>
<th>Cert Type</th>
<th>Minimum Price</th>
<th>Maximum Price</th>
<th>Avg (Std. Dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV</td>
<td>$0</td>
<td>$249</td>
<td>$81 (74)</td>
</tr>
<tr>
<td>OV</td>
<td>$38</td>
<td>$1172</td>
<td>$258 (244)</td>
</tr>
<tr>
<td>EV</td>
<td>$100</td>
<td>$1520</td>
<td>$622 (395)</td>
</tr>
</tbody>
</table>
## Market Share

<table>
<thead>
<tr>
<th>Certificate Type</th>
<th>Market Leaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV</td>
<td>GoDaddy (40%), Symantec/GeoTrust (36%), Symantec/Thawte (10%)</td>
</tr>
<tr>
<td>OV</td>
<td>Symantec (54%), Comodo (21%), Entrust (6%), Network Solutions (5%)</td>
</tr>
<tr>
<td>EV</td>
<td>Symantec (68%), Comodo (7.9%), Godaddy (5.2%)</td>
</tr>
</tbody>
</table>
Observations

• This *should* be a **commodity market:**
  – Browsers do not distinguish between cert providers!
    • Certificates are “perfectly substitutable”
  – Buyers cannot distinguish between more/less secure sellers (CAs)!
    – High fixed costs, (very) low marginal costs
• Expect to see competition on price only
  “race to the bottom”
• **Instead:** price variability, market dominated by large players
  – What gives?
Competition Among CAs

What are CAs’ customers buying?

• Brand reputation
  “Nobody ever got fired for buying Verisign [now Symantec].”

• Additional services
  – E.g., certificate management services

• Some CAs may be “too big to fail”...
Risks to the System - I

The DigiNotar Incident

• DigiNotar, a CA in the Netherlands
  – Served as CA for some Dutch government functions
  – Included as trust root in

• Hacked in July 2011
  – Attacker accessed root CA system and issued a wildcard certificate for google.com
    • Subsequently used in a large-scale MITM attack on 300K users in Iran
    • In the interim 531 certs for 53 domain names were issued
  – Incident did not become public until September 2011
  – After investigation, Dutch government took over DigiNotar

• Removed from browser CA lists shortly after
  – DigiNotar declared bankruptcy
Other Incidents

• Larger CAs have also been hacked
  – Verisign (RSA) breach in 2010, not publically acknowledged until 2012
  – Comodo has reportedly been breached several times

• None have been removed from browser CA lists
  – Some CAs are likely too big to fail.
Outline

1. What are we afraid of?
2. Countermeasures: Securing the Web
3. Public-key Crypto and Certificate Authorities
4. A Look at the “CA ecosystem”
5. Problems and Solutions
Systemic Problems

• Any CA can issue a certificate for any site. That is: trustworthiness of amazon.com’s cert does not depend (only) on the security practices of its issuer. ...also depends on the practices of all other CAs!
  Trustworthiness of the entire system cannot exceed the trustworthiness of its weakest component.

• Information asymmetry abounds.
  – Security practices of CAs are not visible to the stakeholders who are most at risk.
    “There seems to be wide consensus that the average end-user cannot reasonably be expected to exert control over the HTTPS ecosystem.” [1]
  – CAs have strong incentives not to reveal security incidents.

• Risks to some large CAs are externalized.

Potential Solutions

• DANE: DNS-based Authentication of Named Entities
  – Store cert-related information in DNS to increase trust
  – E.g., name of CA authorized to issue certs for amazon.com
  – In the limit: public key info
  – Requires DNSSEC deployment to secure the DNS info

• Convergence, Perspective (convergence.io)
  – Rely on consensus of a set of Notaries to determine reliability of a cert
  – Users set their own policies on which Notaries to trust
  – Anyone can be a trust Notary
Summary

• The current architecture of trust for HTTPS (indeed, anything using SSL/TLS) is broken.
  – Information asymmetry abounds
  – Brand reputation is about the only competitive factor
  – Incentives are unclear, even perverse
    • E.g., browser vendors consider everything in terms of performance
  – Some CAs are “too big to fail”

• Good technological solutions exist.
  – Most involve adding new sources of info/replacing CAs
  – But it will take a while for them to be deployed

• The real question is not “Should I trust the padlock?” but “Do I have a choice?”
More Stuff to Keep you Up at Night

• What should public keys actually be bound to?
  – Domain names?
  – Organizations?
  – People?

• Can you tell the difference between “KINKOS” and “KINKOS”?
  Fourth letter in the first is Unicode 0x4B, LATIN CAPITAL LETTER K; in
  the second it is 0x039A, GREEK CAPITAL LETTER KAPPA
  – What stops me from registering the second one under .com?