CSE 484 / CSE M 584: Computer Security and Privacy

Web Security
[Overview + Browser Security Model]

Spring 2019

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials...
Recall: Authenticity of Public Keys

Problem: How does Alice know that the public key she received is really Bob’s public key?
You encounter this every day...

SSL/TLS: Encryption & authentication for connections
Example of a Certificate

GeoTrust Global CA
Google Internet Authority G2

*google.com
Issued by: Google Internet Authority G2
Expires: Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time
This certificate is valid

Details

Subject Name
Country US
State/Province California
Locality Mountain View
Organization Google Inc
Common Name *google.com

Issuer Name
Country US
Organization Google Inc
Common Name Google Internet Authority G2

Signature Algorithm SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters none

Not Valid Before Wednesday, April 8, 2015 at 6:40:10 AM Pacific Daylight Time
Not Valid After Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time

Public Key Info
Algorithm Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key 65 bytes: 04 CB DD C1 CE AC D6 20 ...
Key Size 256 bits
Key Usage Encrypt, Verify, Derive
Signature 256 bytes: 34 8B 7D 64 5A 64 08 5B ...
Many Challenges...

• Hash collisions

• Weak security at CAs
  – Allows attackers to issue rogue certificates

• Users don’t notice when attacks happen
  – We’ll talk more about this later in the course

• Etc...
Attacking CAs

Security of DigiNotar servers:

• All core certificate servers controlled by a single admin password (Pr0d@dm1n)
• Software on public-facing servers out of date, unpatched
• No anti-virus (could have detected attack)
Consequences

• Attacker needs to first divert users to an attacker-controlled site instead of Google, Yahoo, Skype, but then...
  – For example, use DNS to poison the mapping of mail.yahoo.com to an IP address

• ... “authenticate” as the real site

• ... decrypt all data sent by users
  – Email, phone conversations, Web browsing
 Attempt to Fix CA Problems: Certificate Transparency

• **Problem:** browsers will think nothing is wrong with a rogue certificate until revoked

• **Goal:** make it impossible for a CA to issue a bad certificate for a domain *without the owner of that domain knowing*  
  – (Then what?)

• **Approach:** auditable certificate logs

www.certificate-transparency.org
Attempt to Fix CA Problems:
Certificate Pinning

- **Trust on first access**: tells browser how to act on subsequent connections
- **HPKP – HTTP Public Key Pinning**
  - Use these keys!
  - HTTP response header field "Public-Key-Pins"
- **HSTS – HTTP Strict Transport Security**
  - Only access server via HTTPS
  - HTTP response header field "Strict-Transport-Security"
Keys for People: Keybase

• Basic idea:
  – Rely on existing trust of a person’s ownership of other accounts (e.g., Twitter, GitHub, website)
  – Each user publishes signed proofs to their linked account

Verifying myself: I am franziroesner on Keybase.io. 5YGG83pd-i4zvvl2dUHDMrOouRG386Q_tZ / keybase.io/franziroesner/…

https://keybase.io/
Web+Browser Security
Big Picture: Browser and Network

Browser

- OS
- Hardware

Network

website

request

reply
Where Does the Attacker Live?

Mitigation: SSL/TLS (not covered further)

Mitigation: Browser security model + web app security (today + next week)
Web Attacker

• Controls a malicious website (attacker.com)
  – Can even obtain SSL/TLS certificate for site

• User visits attacker.com – why?
  – Phishing email, enticing content, search results, placed by an ad network, blind luck ...

• Attacker has no other access to user machine!

• Variation: good site honest.com, but:
  – An iframe with malicious content included
  – Website has been compromised
Two Sides of Web Security

(1) Web browser
   – Responsible for securely confining content presented by visited websites

(2) Web applications
   – Online merchants, banks, blogs, Google Apps ...
   – Mix of server-side and client-side code
     • Server-side code written in PHP, Ruby, ASP, JSP
     • Client-side code written in JavaScript
   – Many potential bugs: XSS, XSRF, SQL injection
All of These Should Be Safe

• Safe to visit an evil website

• Safe to visit two pages at the same time

• Safe delegation
Browser Security Model

Goal 1: Protect local system from web attacker
  → Browser Sandbox

Goal 2: Protect/isolate web content from other web content
  → Same Origin Policy
     (plus sandbox)
Browser Sandbox

Goals: Protect local system from web attacker; protect websites from each other

– E.g., safely execute JavaScript provided by a website
– No direct file access, limited access to OS, network, browser data, content from other websites
– Tabs (new: also iframes!) in their own processes
– Implementation is browser and OS specific*

*For example, see: https://chromium.googlesource.com/chromium/src/+master/docs/design/sandbox.md

<table>
<thead>
<tr>
<th>High-quality report with functional exploit [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandbox Escape [5]</td>
</tr>
</tbody>
</table>

From Chrome Bug Bounty Program
### Same Origin Policy

**Goal:** Protect/isolate web content from other web content

Website origin = (scheme, domain, port)

<table>
<thead>
<tr>
<th>Compared URL</th>
<th>Outcome</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.com/dir/page.html">http://www.example.com/dir/page.html</a></td>
<td>Success</td>
<td>Same protocol and host</td>
</tr>
<tr>
<td><a href="http://www.example.com/dir2/other.html">http://www.example.com/dir2/other.html</a></td>
<td>Success</td>
<td>Same protocol and host</td>
</tr>
<tr>
<td><a href="http://www.example.com:81/dir/other.html">http://www.example.com:81/dir/other.html</a></td>
<td>Failure</td>
<td>Same protocol and host but different port</td>
</tr>
<tr>
<td><a href="https://www.example.com/dir/other.html">https://www.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different protocol</td>
</tr>
<tr>
<td><a href="http://en.example.com/dir/other.html">http://en.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host</td>
</tr>
<tr>
<td><a href="http://example.com/dir/other.html">http://example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
<tr>
<td><a href="http://v2.www.example.com/dir/other.html">http://v2.www.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
</tbody>
</table>

[Example from Wikipedia]
Same Origin Policy is Subtle!

• Some examples of how messy it gets in practice...
• Browsers don’t (or didn’t) always get it right...

• We’ll talk about:
  – DOM / HTML Elements
  – Navigation
  – Cookie Reading
  – Cookie Writing
  – Iframes vs. Scripts