How do we get to
TLS Everywhere?

Eric Rescorla
ekr@rtfm.com

IETF 83
Web security depends on communications security

• Most of the Web threat model just assumes traffic confidentiality and integrity
  – Which means HTTPS

• On paper things look pretty good
  – Lots of standards
    * Over 27 TLS WG RFCs
  – Every major browser and server supports SSL/TLS
    * Though browsers tend to run older versions
Actual picture is much worse

- The vast majority of Web traffic is not encrypted [CAP10]
- About 1% of sites even offer HTTPS [KKG⁺10]
- This number should be 100%
What’s going on?

• Certificates are a huge mess
  – Too hard to get for the right people
  – Too easy to get for the wrong people

• Hard to deploy once you have a certificate
  – Mixed content
  – People still type http://

• Performance concerns
  – Real but a distinctly low order bit
Getting a certificate

“I can’t f’ing figure out how to get a cert from go daddy - kid you not
...
god help people that don’t know what a CSR is
...
I am like 45 minutes in ”

Cullen Jennings, PhD
— Cisco Fellow
Former IETF Area Director
Can we dispense with certificates?

- Lots of designs for alternative site authentication schemes
- For example, store the keys in DNS (secured with DNSSEC), aka DANE
- Could we replace X.509 with these?
Alternative certificate systems: a collective action problem

• Current situation: all browsers support X.509/PKIX
  – No browsers support anything else

• Any server which wants to have TLS must have both credentials
  – Until practically all clients support the new system
  – This means clients get no benefit from the new system
    * So little pressure to add client-side support
    * Which means little value in server deployment

• When can a server stop supporting PKIX? Ever?

• Note: this doesn’t apply to systems intended to restrict PKIX certs
Worked Example: Server Name Indication

- Original design of TLS supported one server per IP
  - Even though HTTP allows virtual servers via Host header
  - This makes TLS virtual hosting very expensive with IPv4
- TLS *Server Name Indication* (SNI) fixed this problem in 2003 [BWNH+03]
  - Now supported almost everywhere*
  - ... but not on IE on Windows XP (XP is 30% of market!)
- So still not totally safe to run an SNI-based server

*http://en.wikipedia.org/wiki/Server_Name_Indication
Converting people to HTTPS

• So you’ve turned on HTTPS, now what?
  – Users still type http://

• You could HTTP redirect them to HTTPS site
  – But now you have an active attack/downgrade problem
  – We need this to be as secure/sticky as possible

• Possibilities
  – Redirects + HSTS?
  – SPDY upgrade (but what about HTTP?)
  – DNS records?
  – HTTPS Everywhere?
  – Something else?
Active Mixed Content

- All JavaScript (JS) code on a page runs in the same security context
  - No matter where it was retrieved from
  - And it has access to basically all the page data

- What happens when you load a script over HTTP
  - From an HTTPS page
  - “Active mixed content”

- An attacker can modify the insecure JS
  - And completely owns your page

- Not much better than running everything over HTTP
  - (But still better)
Modern Web pages are full of external scripts
Modern Web pages are full of external scripts
Modern Browsers Don’t Like Active Mixed Content

Hello World
Mixed-Content:
Another collective action problem

• Say I turn on HTTPS for my site
  – But I have all these HTTP dependencies
  – Now everything breaks!

• And it gets worse as browser manufacturers clamp down
  – (But this is an important security feature)

• How do we square incremental deployment with mixed content protection?
  – We want everyone to gradually migrate to TLS
  – But they won’t do it if everything breaks when they turn it on
Summary

• Web security depends critically on communications security
  ... which means TLS

• Needs to be easier to use TLS everywhere
  – Make getting server-side credentials easier
    ... and harder for attackers
  – Making it safe to turn on HTTPS on your own site
    ... even in the face of mixed content
  – Automatically converting HTTP users to HTTPS users
    ... as securely as possible
References

