Overall Agenda

The Current Web Has Some Holes

Help is On The Way (Some Anyway)

Solving Real World Problems

Still A Ways To Go
The Current Web Has Some Holes
Sort of Like This:
It Is Hard To Do Even Simple Things Safely

- Include an ad on your site
- Use third-party Site-Analytics
- Allow user input ("Rich" or otherwise)
- Uniform use of HTTPS
What Are Some of the Holes?

- Cross Site Request Forgery (CSRF)
- Cross Site Scripting (XSS)
- Clickjacking
- Malvertising
- TLS/SSL Man In The Middle (MITM)
- For example - sslstrip
Why Do These Attacks Exist

- Core protocol/technology weaknesses
- Too much required of each and every developer
- Lack of Security Policy Mechanisms
Core Protocols/Technologies Have Weaknesses

• Cookies are broken:
  • Their scope rules are broken
  • “Secure” Flag doesn’t really mean the same thing everywhere
  • “HTTPOnly” and “Secure” only partially effective
  • Network MiTM attacker can overwrite cookies by spoofing..
    http://www.example.com
    .. to overwrite real “secure cookies” for..
    httpS://www.example.com

• Practically anything can be interpreted as JavaScript

• Browsers default to HTTP first (Not HTTPS)
Too Much Required of Each and Every Developer

• To Implement a “Strong” Security Policy……

• Every Cookie has to have HTTPonly and Secure Flag

• Every link generated has to have the right scheme (HTTP vs. HTTPS)

• Every page must have the right content encoding

  • This is TOO HARD
Lack of “Site” Security Policy Mechanisms

- A Developer or WebSite Administrator has *no coherent way* to say, for example:
  - Treat all my cookies “Securely”,
  - Only load HTTPS Content,
  - And don’t frame my site.
Help is On The Way
(Some Anyway)
Web Security Today

However, help is on the way...
Help is on the Way

Though, it might not be quite what you want....
“Emerging” Web Security Standards

- X-Frame-Options
- HTTP Strict-Transport-Security (HSTS)
- Mozilla Content Security Policy (CSP)
- NoScript Application Boundary Enforcer (ABE)
- Cross Origin Resource Sharing (CORS)
- X-Content-Type-Options: nosniff
Some Details On...

X-Frame-Options

HTTP Strict Transport Security

Content Security Policy (CSP)
Control Who Can Frame Your Site (X-Frame-Options)

- Doing this in JavaScript is an exercise in futility
- X-Frame-Options (Invented by Microsoft In 2009)
- HTTP header that tells a browser whether to allow framing (and by whom)
  - Now widely implemented
  - Not very flexible
  - Doesn’t solve all use cases

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
Date: Fri, 07 Jan 2011 17:38:01 GMT
X-Frame-Options: SAMEORIGIN
```
Force Your Entire Site to HTTPS (HTTP Strict Transport Security)

- **ForceHTTPS** conceived by Jackson and Barth in 2007
  - In response to others' approaches (e.g. Locked-Same-Origin)
- Presented at WWW 2008 (April) [1]
- General notion discussed sporadically since publication
- Initially spec was known as **ForceTLS**
- Presently a draft specification at the IETF

Force Your Entire Site to HTTPS (HTTP Strict Transport Security)

• Ostensibly simple high-level use case:
  • *Web browser users* wish to use sites securely
  • *Web site deployer* wishes to offer site securely

• STS Server declares STS policy by returning STS response header:

  HTTP/1.1 200 OK
  Cache-Control: private
  Pragma: no-cache
  **Strict-Transport-Security**: max-age=31536000

• Helps mitigate attacks such as sslstrip and malicious network operators
Content Security Policy (CSP)

• Mozilla developed CSP to help prevent common web attacks such as XSS

• Side Benefit - Clickjacking Protection

• Allows a site to specific how and from where content (scripts, images, css, etc) will load/execute.
What Can You Do With CSP?

- Force all JavaScript to load from include file, not inline
- Eliminate certain JavaScript usage (e.g. "eval()")
- Whitelist where JS can be loaded from (eliminate injections)
- Detect violations of your defined web security policy
- Control who can frame your content
- And much more…

HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
X-Content-Security-Policy: allow 'self'; img-src *;
object-src medial1.com media2.com *.cdn.com; script-src trustedscripts.example.com
Solving Real World Problems
Solving Real World Problems: Agenda

Prevent Framing of Your Site

Make All Your Cookies “Secure”

Make Your Entire Website HTTPS Only
Prevent Framing of Your Site

• Harder than it looks – should do ALL of these:
  • Deploy “Framebusting” JavaScript
    • Though, most are relatively easily defeated (see [2])
  • Deploy X-Frame-Options Headers
    
    HTTP/1.1 200 OK
    Content-Type: text/xml; charset=UTF-8
    Date: Fri, 07 Jan 2011 17:38:01 GMT
    X-Frame-Options: DENY

• Note: Not supported in older browsers!

• For Clients running Firefox-4
  • Use CSP to control frame-ancestors

Make All Your Cookies “Secure”

HTTP/1.1 200 OK
Date: Fri, 07 Jan 2011 17:52:34 GMT
Server: Apache
Set-Cookie: Name=Value; domain=paypal.com; path=/;

- Lacking “HTTPOnly” and “Secure” flags
Make All Your Cookies “Secure”

HTTP/1.1 200 OK
Date: Fri, 07 Jan 2011 17:52:34 GMT
Server: Apache
Set-Cookie: Name=Value; domain=paypal.com; path=/; Secure; HttpOnly

- No “standard approach” that makes this easy for you
- Web Application Firewalls or Servlet Filters can help
- E.g. if using Apache server, then use ModSecurity to...
  - Craft ruleset that checks for cookies lacking either/both flags and adds them if needed [3]
Make Your Entire Website HTTPS Only

- Convert your entire website to **serve all content over HTTPS**
  - Convert all links you generate to “https:”
  - Beware of content served from other domains!
    - e.g. Images you include, web analytics Javascript
    - Make sure you redirect from HTTP to HTTPS with an HTTP-301 (SEO reasons)

- And, deploy HSTS headers:

```plaintext
HTTP/1.1 200 OK
Date: Fri, 07 Jan 2011 17:19:03 GMT
Server: Apache
Cache-Control: private
Pragma: no-cache
Strict-Transport-Security: max-age=31536000
```
Still a Ways To Go
Still a Ways to Go...

• Remember that list of things you couldn’t do safely?
  • Include an ad on your site
  • Use third-party Site-Analytics
  • Allow user input (“Rich” or otherwise)
  • Uniform use of HTTPS
  • etc.

• You still can’t in some cases
Why? Some Current Constraints...

• Lack of standards for general purpose *Security Policy mechanisms*

• Can’t safely frame third-party content directly without lots of “attack surface”

• Can’t safely embed third-party script

• Not all deployed browsers implement the new security features/standards

• And also because...
  • You can't “break the Web”
Some Goals for Approaches

• Should not rely on every developer (and user) “getting it right” 100% of the time

• Security mechanisms should be “declarative policy and configuration”
  • separate from “code”

• Reduce the need for new individual HTTP headers for each specific issue

• Overall – create security mechanisms that allow/enforce the concept of **Least Privilege**
How Do We Get Started

• What we need..

  • Commonly agreed to terms and definitions of Web security concepts
    • e.g. “origin”, “site”

  • Web Security Policy Framework
    • and coherent notions of what that *means*

• Forums in which to have the discussion
Some Work In Progress

• IETF WebSec Working Group

• W3C WebAppSec and WebApps Working Groups

• Related:
  • IETF DANE WG (a.k.a. “Keys in DNS (KIDNS)” and “KeyAssure”)
Some Work In Progress

- IETF WebSec WG
  - Newly-established: 12-Oct-2010
  - Initial existing specifications...
  - HSTS – HTTP Strict Transport Security
  - Origin definition and explicit header
    - [draft-ietf-websec-origin](https://tools.ietf.org/pdf/draft-ietf-websec-origin)
  - Content Sniffing Rules
    - [draft-ietf-websec-mime-sniff](https://tools.ietf.org/pdf/draft-ietf-websec-mime-sniff)
  - Will develop...
    - HTTP app security "problem statement and requirements" doc
Some Work In Progress

- W3C WebAppSec and WebApps Working Groups
  - WebAppSec WG
    - Nascent
    - Proposed charter: [http://www.w3.org/2010/07/appsecwg-charter.html](http://www.w3.org/2010/07/appsecwg-charter.html)
  - Mission:
    - “...to develop security and policy mechanisms to improve the security of Web Applications, and enable secure cross-site communication.”

- Have security-oriented specs from WebApps move to WebAppSec...
  - Cross-Origin Resource Sharing (CORS)
  - Uniform Messaging Policy (UMP)
  - Content Security Policy (CSP) (from Mozilla, yet to appear as a W3C draft)
  - (Security-oriented portions of HTML5?)

- WebApps working group: [http://www.w3.org/2008/webapps/](http://www.w3.org/2008/webapps/)
Some Work In Progress

- IETF DANE WG
  - “DNS-based Authentication of Named Entities”
    - (a.k.a. “Keys in DNS (KIDNS)”, nee “KeyAssure”)
- Objective:
  - “Specify mechanisms and techniques that allow Internet applications to establish cryptographically secured communications by using information distributed through DNSSEC for discovering and authenticating public keys which are associated with a service located at a domain name.”
- Draft specification:
  - Using Secure DNS to Associate Certificates with Domain Names For TLS
    - [draft-ietf-dane-protocol](https://tools.ietf.org/draft-ietf-dane-protocol)
Some Work Still Lacking a Home

- Common Security User-Interfaces
  - Browsers presently display security issues differently
  - Also have differing approaches to dealing with issues

- Fixing the Certificate Authority (CA) Situation
  - Multitude of CAs in browser & OS “Trust Anchor Repositories (TARs)”
  - All trusted equally
  - Each can certify any domain name
  - Large attack surface
  - CA/Browser Forum + WebTrust ?
What Can You Do to Help?

- Participate in the IETF and W3C Working Groups
- Deploy your website uniformly via HTTPS
- Deploy HSTS and CSP on your website
- Provide feedback to the working groups
Web Security Tomorrow (our desire)
Questions?

• For more details:

  • The Need for Coherent Web Security Policy Framework(s)

  • Jeff Hodges (Jeff.Hodges@paypal.com)
  • Andy Steingruebl (asteingruebl@paypal.com)