Securing the Web Platform

Collin Jackson
collin.jackson@sv.cmu.edu
Web: The OS of the Future?

- Dynamic Interactive
- Ubiquitous Instant updates

Pages → Web Applications → Programs
Remote code? Are you crazy??

- **Integrity**
  - Compromise your machine
  - Install a malware rootkit
  - Buy stuff with your credit card

- **Confidentiality**
  - Steal passwords
  - Read your email
Browser Sandbox

- **Goal**
  - Run remote web applications safely
  - Limit access to OS, network, and browser data

- **Approach**
  - Isolate sites in different security contexts
  - Browser manages resources, like an OS
What the Sandbox Can't Stop

Cross-Site Scripting (XSS)

Clickjacking

Cross-Site Request Forgery (CSRF)

Network Attacks (Firesheep, etc.)
Browser Vulnerability Mitigation

- Vulnerabilities typically affect complex components of the browser
- Sandbox contains exploits and mitigates damage
- Protect local file system from theft and modification
- Keep web malware isolated from trusted sites (?)
### Web Vulnerability Mitigation

![Table](browserscope.org)

Source: browserscope.org
Observing Browsers in the Wild

- $100 = 174,250 impressions
- Observe browsers in the wild
- Target interesting populations
- Many applications
  - DNS rebinding
  - Referrer stripping
  - Patch deployment
  - HTTPS tripwires
  - …
Talking to Yourself for Fun and Profit

- Cloud applications need to talk to the cloud!
- Raw sockets are most efficient
Existing Approaches

- **Java: Only talk to yourself**
  ```java
  new java.net.Socket(host, port);
  ```

- **Flash Player: Policy file**
  ```xml
  <policy-file-request/>
  <allow-access-from domain="*" />
  ```
Transparent Proxies

- Inspect and modify HTTP traffic
  - Caching
  - Filtering
  - Compression
IP Hijacking

- Host routing allows connection to unauthorized destinations
Cache Poisoning

- IP routing and Host caching is much worse

Alice
Java VM
attacker.class

Bob
Java VM
attacker.class

Proxy
Cache miss

Route by IP

GET /script.js HTTP/1.1 Host: www.google.com IP -> 2.2.2.2

Cache

Cache by Host

GET /script.js HTTP/1.1 Host: 1.1.1.1 IP -> 1.1.1.1

attacker.com
IP: 2.2.2.2

www.google.com
IP: 1.1.1.1

script.js

script.js

Cache hit

GET /script.js HTTP/1.1 Host: www.google.com IP -> 1.1.1.1

script.js

Observed Vulnerabilities

- **IP Hijacking**
  - Java: 3,152 of 51,273 (6.1%)
  - Flash Player: 2,109 of 30,045 (7%)

- **Cache poisoning**
  - Java: 53 of 30,045 (0.18%)
  - Flash Player: 108 of 51,273 (0.21%)
  - One successful cache poisoning attack per $0.93
  - Affects ALL clients of the transparent proxy
Reactions

Chromium Code Reviews

Issue 5643005: Disable WebSocket by default

Description

Disabling the WebSocket protocol
8th December 2010

Adam Barth reported on vulnerabilities with transparent/intercepting proxies affecting a user getting a JavaScript file from an attacker. This means that until the new WebSocket protocol is fixed, it is true for Firefox.

Encodings: US-ASCII

Copyright © 2003-2011 Anne van Kesteren

WebSocket disabled in Firefox 4

on December 8, 2010 by Chris Heilmann

This is a serious threat to the Internet and WebSocket and not a browser specific issue. The protocol vulnerabilities also affect Java and Flash solutions. In a web environment that could for example mean that a widely used JavaScript file – like Google analytics – could be replaced on a cache that is served in the browser and the Internet.

No WebSocket support in Firefox 4 and Opera until the security issues are fixed
New Handshake

- Prevent attacker from controlling bytes sent by the client

CONNECT websocket.invalid:443 HTTP/1.1
Host: websocket.invalid:443
Sec-WebSocket-Key: <connection-key>
Sec-WebSocket-Metadata: <metadata>

HTTP/1.1 200 OK
Sec-WebSocket-Accept: <hmac>

<Masked messages>
<Plaintext messages>
Performance

(a) 10 byte data frames

(b) 100 byte data frames

(a) 1,000 byte data frames
Future Work

- Masking for Java & Flash
- Identifying affected population
History Hijacking

Nosy website

Victim
History Hijacking

Nosy website

Victim

Facebook

Bank of America

YouPorn
History Hijacking in Practice

- Determine whether user has visited any URL
- Used by many top sites
  - ~200 queries each
- New browser defense
  - Prevent JavaScript from determining link color
  - Ban other styling besides color
Interactive Attacks: Chessboard

Please click on all of the chess pawns.

Each pawn is a link to a different site.
Interactive Attacks: CAPTCHA

Please type the string of characters shown below, then press RETURN. You don’t have to match upper and lower case.

FA4A SABA A-65 A9-5
How it works
Queries per minute

- Word CAPTCHA
- Char. CAPTCHA
- Chessboard
- Pat. match
- Auto (direct)
- Auto (indirect)
- Auto (timing)
Automated Attacks
One Possible Defense

Private Browsing

Firefox won’t remember any history for this session.

In a Private Browsing session, Firefox won’t keep any browser history, search history, download history, web form history, or temporary internet files. However, files you download and bookmarks you add will be kept.

You’ve gone incognito. Pages you view in this window won’t appear in your browser history or search history, and they won’t leave other traces, like cookies, on your computer after you close the incognito window. Any files you download or bookmarks you create will be preserved, however.

Going incognito doesn’t affect the behavior of other people, servers, or software. Be wary of:

- Websites that collect or share information about you
- Internet service providers or employers that track the pages you visit
- Malicious software that tracks your keystrokes in exchange for free smilies
- Surveillance by secret agents
- People standing behind you

Learn more about incognito browsing.

![Chart showing usage in different browsers]
Future Work

- History sniffing is here to stay
  - As long as history is visible on screen, user can unintentionally leak it
- Effective defenses have significant usability costs
- Regulatory defenses may be more effective than technical ones
  - See also: Do Not Track
App Isolation for Improved Security

Collaborators: Adam Barth (Google), Jason Bau (Stanford), Eric Chen and Collin Jackson (CMU)

- “Use a separate browser for banking”
- Prevents common web attacks
  - Harder to get user to visit malicious URL
  - Harder to hijack session
- … but hard to use
  - Must do it every time
  - No unified browser interface
Can we do this automatically?

- Application declares "safe" landing page
- Browser creates process for the app
- Other sites are sandboxed and cannot affect this process
- Automatic, invisible
- Can now multitask within a single browser
Evaluation

- Check design using Alloy, a formal model checking language
  - Described our system using type signatures and relations
  - Stated our constraints as facts
  - Described the end security goals as assertions
- Alloy told us whether our facts were consistent with our assertions.
  - Repaired an inconsistency in our design
collin.jackson@sv.cmu.edu
http://websec.sv.cmu.edu/