Building a Safer Web:
Web Tripwires &
A New Browser Architecture

Charles Reis
University of Washington
Web Browsing isn’t Safe

Install Malware

Exploit

Injected Ads

Send Spam

Modify account

CSRF

Injected Ads

Web Mail

Movie Rentals
This Talk:

- Focus on one problem: *in-flight page changes*
  - Recent study shows undesirable changes
  - Publishers can detect with Web Tripwires

- Broader view of *safe browser architectures*
  - On-going research at UW CSE
1. In-Flight Page Changes & Web Tripwires

Joint work with Steve Gribble, Yoshi Kohno, Nick Weaver
ISP-Injected Ads

- Surprising reports of web page modifications
- How often does this occur?
Detecting Page Changes

- Can detect with JavaScript

- Built a Web Tripwire:
  - Runs in client’s browser
  - Finds most changes to HTML
  - Reports to user & server
How it Works

- Fetch and render original page
- Fetch JavaScript code in background
  - Second, encoded copy of page
  - Can’t compare against DOM directly
  - Use XMLHttpRequest to fetch page’s source code as a string

http://vancouver.cs.washington.edu
Attracting Visitors

✦ Wanted view of many clients on many networks

✦ Posted to Slashdot, Digg, etc.

✦ Visits from over 50,000 unique IP addresses
Outline

Detecting In-Flight Changes

Measurement Results

Dangerous Consequences

Web Tripwires for Publishers
Many Users Affected

- 657 clients saw changes (1.3%)
  - Many made by client software
  - Some made by agents in network
- Diverse incentives
- Often concerning for publishers

http://vancouver.cs.washington.edu
Many Types of Changes

- Internet Service Providers
- Enterprise Firewalls
- Client Proxies
- Malware

http://vancouver.cs.washington.edu
Changes by ISPs

- **Injected Advertisements** *(2.4%)*
  - NebuAd, MetroFi, LokBox, ...

  *Revenue for ISP; annoy users*

- **Compression** *(4.6%)*

Growing Trend?
PerfTech, Front Porch, Adzilla, Phorm

http://vancouver.cs.washington.edu
Changes by Enterprises

- Security Checking Scripts (2.3%)
- BlueCoat Web Filter

Safer for clients; reduce risk

http://vancouver.cs.washington.edu
Changes by Client Proxies

- Popup & Ad Blockers (71%)
- Zone Alarm, Ad Muncher, ...

Less annoying; impact revenue

http://vancouver.cs.washington.edu
Changes by Malware

Server
ISP
Firewall
Client

✦ Adware (1 client)

http://vancouver.cs.washington.edu
Changes by Malware

- **Adware** (1 client)
- **Worms** (2 clients)

Helps malware author; risk to user

http://vancouver.cs.washington.edu
Outline

Detecting In-Flight Changes

Measurement Results

Dangerous Consequences

Web Tripwires for Publishers
Unanticipated Impact

- Some changes *inadvertently* broke pages
- JavaScript errors
- Interfered with MySpace / forum posts
Introduced Vulnerabilities

✦ **XSS** allows script injection
  ✦ Usually fixed at server
  ✦ Some proxies made otherwise safe pages vulnerable
    ✦ Ad Muncher, Proxomitron
  ✦ Affected most HTTP pages
    ✦ Like a root exploit

http://vancouver.cs.washington.edu
XSS via Proxy

- Proxy injected script code
- Page URL was included in code
- Attacker could place script code in a valid URL
- Users who follow the URL run injected code

http://usbank.com/?</script><script>attackCode...
Example Exploit

- Redirect user to Google
- Inject script code into search form
- Append exploit code to all outgoing links

```html
www.usenix.org/events/sec07/wips.html?<script>alert('attack')<script>attackCode...
```
Vulnerability Aftermath

- Reported vulnerabilities; now fixed
- Web tripwires can help find vulnerabilities
  - Search for URL in page changes
How to React?

- Option 1: Use HTTPS
  - Encryption prevents in-flight changes
  - But... costly and rigid
  - Can’t allow security checks, caching, etc.
Web Tripwires

- JavaScript code to detect changes
- Easy for publishers to deploy
  - Configurable toolkit
  - Web tripwire service
- But... not cryptographically secure
- Can be robust in practice

http://vancouver.cs.washington.edu
# Tradeoffs

<table>
<thead>
<tr>
<th>HTTPS</th>
<th>Web Tripwires</th>
</tr>
</thead>
<tbody>
<tr>
<td>✦ Prevents in-flight changes, as well as some useful services</td>
<td>✦ Detects most in-flight changes</td>
</tr>
<tr>
<td>✦ Cryptographically robust</td>
<td>✦ Could face an arms race</td>
</tr>
<tr>
<td>✦ Expensive: certificates, computation, extra RTTs</td>
<td>✦ Obfuscation can challenge adversaries</td>
</tr>
<tr>
<td></td>
<td>✦ Inexpensive to deploy</td>
</tr>
</tbody>
</table>
Performance Impact

- Relative to HTTPS, web tripwires have:
  - Low latency
  - High throughput

http://vancouver.cs.washington.edu
Web Tripwire Summary

- HTTP web pages are being changed in flight
  - Real negative impact for publishers & users
  - Page rewriters have dangerous power
- Web tripwires can help publishers react

http://vancouver.cs.washington.edu
2. Safe Browser Architectures

Joint work with
Steve Gribble, Hank Levy
How did we get here?

- Web content has evolved
- Browser now analogous to OS
- Current architectures inadequate
Safety Threats

- Many more than in-flight page changes
  - Exploits, XSS, CSRF, interference
- Need better support for web programs
  - Must improve both program definitions and browser architectures
Outline

- Defining program boundaries
- Preventing unwanted code
- Isolating programs in browser
- Applying uniform policies

Reis, Gribble, Levy [HotNets ’07]
Can’t identify program boundaries

- Same Origin Policy provides current boundaries
- Flawed approach:
  - Too narrow
  - Too broad
  - Easily compromised
Program Boundaries

- New abstractions:
  - Web program
  - Program instance
- Must explicitly assign resources to programs
Keys as Boundaries

- Author holds a private key
- Web program:
  - Public key
  - Set of signed documents
- No PKI required
Outline

- Defining program boundaries
- Preventing unwanted code
- Isolating programs in browser
- Applying uniform policies
Can’t prevent unwanted code

- Scripts injected via user input (XSS)
- Scripts injected in-flight
Authorized Code

- Need to authorize all web program code

- **Script Whitelists** are a start
  
  *Jim, Swamy, Hicks [WWW '07]; Reis, Gribble, Bershad, Levy*

- Browser ignores any script whose hash is not in list

- Should apply to all active code; could sign whitelist

- Challenges for dynamic pages
Outline

- Defining program boundaries
- Preventing unwanted code
- Isolating programs in browser
- Applying uniform policies
Can’t isolate programs in browser

- Can abuse credentials of other sites (CSRF)
- Failures, resource contention
Program Isolation

- **Privacy:**
  - Isolate credentials between instances

- **Robustness:**
  - OS process for each program instance

*Reis et al. [UW Tech Report ’07]*
Outline

- Defining program boundaries
- Preventing unwanted code
- Isolating programs in browser
- Applying uniform policies
Can’t apply uniform policies

- Each content type has its own security model
- No restrictions on browser extensions
- Can’t reason about a web program’s abilities

Greasemonkey
AdBlock
Java
Flash
Silverlight

44
BrowserShield

Reis, Dunagan, Wang, Dubrovsky, Esmeir [OSDI ’06]

- Interpose on JavaScript code
- Prevent exploits of known vulnerabilities
- Rewrites JavaScript in-flight
- Challenges: HTTPS, other active content, browser quirks
Apply Uniform Policies

- Need to interpose on web content within the browser
- Enforce same policies on all content types
- Protect key resources (DOM, FS, network)
Conclusion

- Many threats in today’s web
  - In-flight page changes pose risks
- Web Tripwires can help detect changes
- Safer browser architectures are needed
  - Program boundaries, authorized code, isolation, uniform policies