Mitigating Malware

Collin Jackson
CS142 – Winter 2009
Approach

- Fact: Browsers will always have bugs
- Goal: Reduce the harm

Frequency of interactions with attacker × Percentage of time vulnerability is unpatched × Damage if attack works

= Harm
Outline

1. Preventing the Introduction
2. Vulnerability Response
3. Failure Containment
PREVENTING THE INTRODUCTION

- Frequency of interactions with attacker
- Percentage of time vulnerability is unpatched
- Damage if attack works
Drive-by downloads

- Silently installs software when web page is loaded
- Increase exposure by compromising other sites and insert code into them
- Sites owners unaware they are participating in an attack

*Provos et al. "All your iFRAMES Point to Us"*
World of Warcraft keylogger

- Flash Player exploit used to install keylogger
- Links to malicious SWF posted on forums

"Solution": Disable hyperlinks on forum
Scaling it up to the entire web

- 1.3% of the incoming search queries to Google returned at least one malware site
- Visit sites with an army of browsers in VMs, check for changes to local system
- Indicate potentially harmful sites in search results
Now do it in the browser

Reported Attack Site!

This web site at azerbaijan.usaid.gov has been reported as an attack site and has been blocked based on your security preferences.

Attack sites try to install programs that steal private information, use your computer to attack others, or damage your system.

Some attack sites intentionally distribute harmful software, but many are compromised without the knowledge or permission of their owners.

Get me out of here!  Why was this site blocked?
Helping the webmaster out

Safe Browsing
diagnostic page for azerbaijan.usaid.gov

What is the current listing status for azerbaijan.usaid.gov?
Site is listed as suspicious - visiting this web site may harm your computer.
Part of this site was listed for suspicious activity 1 time(s) over the past 90 days.

What happened when Google visited this site?
Of the 73 pages we tested on the site over the past 90 days, 17 page(s) resulted in malicious software being downloaded and installed without user consent. The last time Google visited this site was on 2009-03-08, and the last time suspicious content was found on this site was on 2009-01-30.
Malicious software includes 22 adware(s), 7 scripting exploit(s), 1 trojan(s). Successful infection resulted in an average of 0 new processes on the target machine.
Introductions are easy

• Impressions are cheap ($1 = 2000)
• Ad that is harmless today may be malicious tomorrow
• Possible mitigations:
  <iframe security=restricted>
  <iframe sandbox>
VULNERABILITY RESPONSE

Frequency of interactions with attacker

Percentage of time vulnerability is unpatched

Damage if attack works
Closing the vulnerability window

- Delay publication
  - Coordinate with security researchers
  - Offer prizes for responsibly disclosed security bugs
- Make patch available faster
- Deploy patch faster
Obstacles to patch deployment

- Interrupts work flow
- Requires administrator privileges
- Risk of breaking things
- Separate update mechanisms

- Silent approach: GoogleUpdate.exe
Getting better, but not fast enough

Frei et al. Examination of vulnerable online Web browser populations and the "insecurity iceberg"
Announcements

• Office hours have moved
FAILURE CONTAINMENT

- Frequency of interactions with attacker
- Percentage of time vulnerability is unpatched
- Damage if attack works
Protected Mode IE

- IE7 in Vista is a "low rights" process
- Can prompt user to get more privileges
IE7 Containment Goals

• Arbitrary code execution won't let attacker:
  – Install software
  – Copy files to startup folder
  – Change homepage or search provider setting

• Can we do more?
Containment Goals

Universal XSS
File Theft
Arbitrary Code Execution
Chromium Security Architecture

• Browser ("kernel")
  – Full privileges (file system, networking)
  – Coarse-grained security policies protect local system
• Rendering engine
  – Sandboxed
  – Fine-grained same origin policy enforcement
• One process per plugin
  – Sandboxing optional

Barth et al. "The Security Architecture of the Chromium Browser"
Preventing File Theft

- File Downloads.
  - Renderer can only write files to My Documents\Downloads

- File Uploads.
  - Renderer is granted ability to upload file using browser kernel's file picker.

- Network Requests.
  - Can only request web-safe schemes (http, https, ftp)
  - Dedicated renderers for file://
## Task Allocation

<table>
<thead>
<tr>
<th>Rendering Engine</th>
<th>Browser Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML parsing</td>
<td>Cookie database</td>
</tr>
<tr>
<td>CSS parsing</td>
<td>History database</td>
</tr>
<tr>
<td>Image decoding</td>
<td>Password database</td>
</tr>
<tr>
<td>JavaScript interpreter</td>
<td>Window management</td>
</tr>
<tr>
<td>Regular expressions</td>
<td>Location bar</td>
</tr>
<tr>
<td>Layout</td>
<td>Safe Browsing blacklist</td>
</tr>
<tr>
<td>Document Object Model</td>
<td>Network stack</td>
</tr>
<tr>
<td>Rendering</td>
<td>SSL/TLS</td>
</tr>
<tr>
<td>SVG</td>
<td>Disk cache</td>
</tr>
<tr>
<td>XML parsing</td>
<td>Download manager</td>
</tr>
<tr>
<td>XSLT</td>
<td>Clipboard</td>
</tr>
</tbody>
</table>

### Both

- URL parsing
- Unicode parsing
Is the "kernel" too complex?

- Total CVEs:

<table>
<thead>
<tr>
<th></th>
<th>Browser</th>
<th>Renderer</th>
<th>Unclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>4</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Firefox</td>
<td>17</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td>Safari</td>
<td>12</td>
<td>37</td>
<td>1</td>
</tr>
</tbody>
</table>

- Arbitrary code execution vulnerabilities:

<table>
<thead>
<tr>
<th></th>
<th>Browser</th>
<th>Renderer</th>
<th>Unclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>1</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Firefox</td>
<td>5</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>Safari</td>
<td>5</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
OP Browser

- Fine-grained componentization
- Want to mitigate UXSS
- Focus is on plugin containment
  - Will plugins refuse to be contained?
  - Historically a platform for innovation in policy
- Missing a basic issue...

Grier et al. "Secure web browsing with the OP web browser"
Why UXSS Containment is Hard

Both requests carry cookies!
Tahoma's Approach

- Very coarse grained policy
- Separate browser state for each top-level site
- Site can opt in to more sharing via manifest files

Cox et al. "A Safety-Oriented Platform for Web Applications"
Gazelle's Approach

- Inspect cross-origin HTTP responses
- Filter unexpected content types

Wang et al. "The Multi-Principal OS Construction of the Gazelle Web Browser"
Another approach: Cookie Blocking

- Block the "Cookie" header for cross-domain resource loads
- Third-party cookie blocking already does this for privacy
- Third-party frames are ok
- Cross-subdomain might be ok

Open question: How many sites does this break compared to content type filtering?
Conclusion

Frequency of interactions with attacker

Percentage of time vulnerability is unpatched

Damage if attack works

1. Preventing the Introduction
2. Vulnerability Response
3. Failure Containment
Reading

• Barth et al. "The Security Architecture of the Chromium Browser"

• Optional (i.e. not required):
  – Provos et al. "All your iFRAMES Point to Us"
  – Frei et al. Examination of vulnerable online Web browser populations and the "insecurity iceberg"
  – Cox et al. "A Safety-Oriented Platform for Web Applications"
  – Grier et al. "Secure web browsing with the OP web browser"
  – Wang et al. "The Multi-Principal OS Construction of the Gazelle Web Browser"