Browser Based Defenses

Introducing x06d

james@bluenotch.com

The Problem: Re-Anonymizing You!

- Overall State of the Web
  - Client/Browser issues
    - Hard to make public browsers secure
    - …at least enough to keep the public safe
  - Server landscape
    - Many layers to secure
    - Portions of a served app tend to be clients of another site (see above)
- Well put by RSnake and Jabra's in "De-Anonymizing You!" (DEF CON 17)
- x06p is POC for a Browser Based Defense
- Goal : buy time for 6 days of pseudo protection

Defending Technique: SARS!

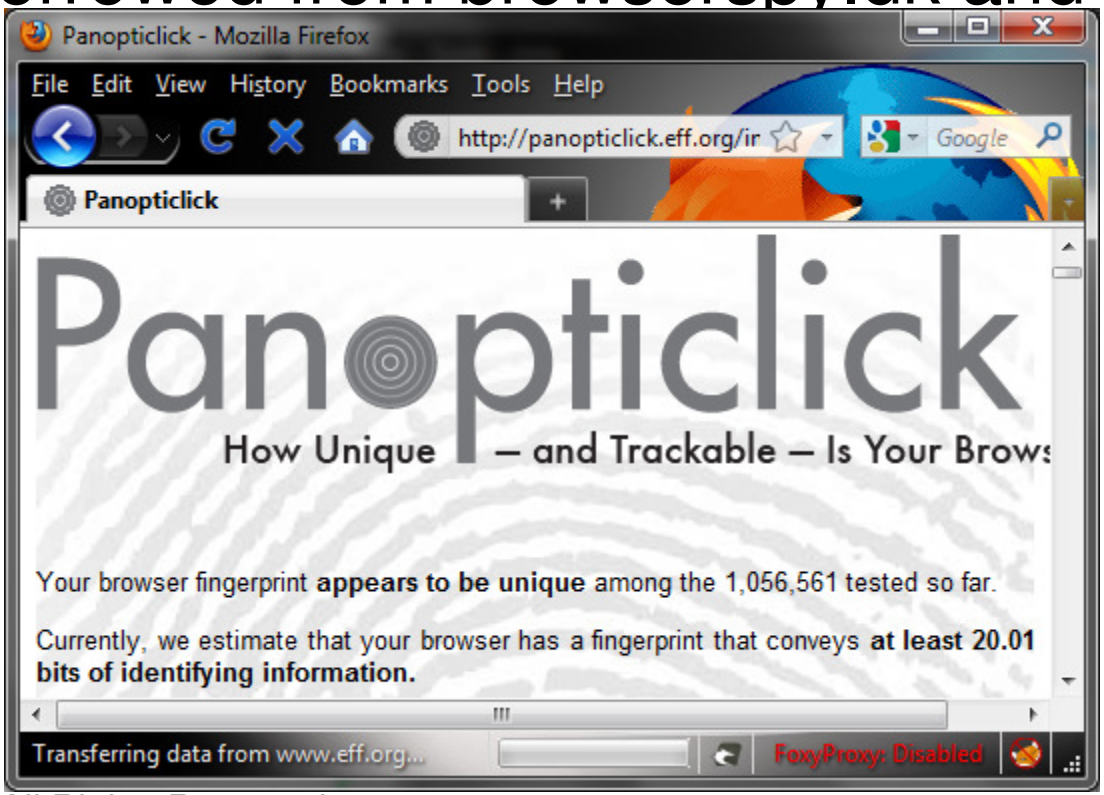
- Sanitize input to the Browser
  - Detect interesting code into the browser
  - Allow control of server content (ala noscript for <script>)
- Anonymize the Browser
  - Make yourself look like everybody else
  - Make yourself look like a specific somebody
- Randomize the Behavior
  - Create a generic history
  - Generate line noise
- Sanitize output to the Server
  - Detect interesting code sent by the browser (ie. XSS)
  - Neuter interesting code (convert the code)

Defeating Attacks on Input

- Sanitize input to the Browser
  - Scan the HTTP Response for evil
  - Plug-ins like noscript already do this
- Whitelisting is hard
  - Site content changes
  - Who is qualified to OK content?
  - Dare we vote on each <script> tag?
- A public blacklist will help
- Might as well live with signature detection shim

Browser Tracking

- panopticlick.eff.org
  - Some code borrowed from browserspy.dk and breadcrumbs
  - Headers
  - History
  - Fonts
  - Plug-ins



Defeating Header Detection

- Generalize every Request Headers except for the URL and HOST
- Randomize parameters to increase noise
  - Change order of GET/POST parameters
  - Add benign parameters
- Cookie Automation for privacy
  - Clear on browser open
  - Rewrite the cookie when stored, put back before use
  - Clear on browser close

Browser Tracking Defenses

- Easiest: be just like everybody else
- Possible: be like somebody you want to frame
- SARS everything in your browser
  - Fake it dynamically
  - Set it up before browsing

Defeating Font Detection

- Install/remove system fonts until you match everyone else
  - Easier in a VM with no third party apps
  - Time/Bandwidth/Storage costs
- Install the same fonts as one specific user
  - Make a browser snapshot
  - Share or trade them?

Defeating History Enumeration

- Go everywhere, at least Alexia top 500
  - Automate and forget
  - Instead of clear, edit history to top 500
- HEAD of FAVICON not good enough
  - HEAD of everything on /index.* ←likely ok
  - Re-crawl the history in the background slowly helps to hide new patterns
  - Comb history removing non-popular sites when not it use

Defeating Font and Plug-in Detection

- Similar to history, but no base-line
  - Figure out a “normal” configuration, and groom the browser to stay that way
  - Stash non-standard fonts while browsing, replacing when done
    - Problematic for multitasking
    - Possible to get stuck if a page installs one
- Plug-ins are hard because of browser versions

New Problems with Plug-ins

- What if the farmville plug-in is popular
  - Facebook users should not dictate “normal”
  - Can we hook the browser, hiding plug-ins?
  - Can we create a benign trojan plug-in to use?

Defeating Attacks on Output

- XSS Browser Helper Objects exist already
  - Trap the Request sent by the browser
  - Scan for HTML action tags
    - <script>, <iframe>, <form>, etc.
    - Problematic for CMS where you want this
    - Tough to normalize reliably
- Supplement with shenanigan detection
  - Compare the output with other browsers
  - Frequent false positives
  - Use a rating ala SpamAssassin's

Font Defense Details

- Detecting fonts is easy with Flash or Java
  - TextField.getFontList();
  - java.awt.GraphicsEnvironment.getAllFonts();
- Requires more work in Javascript
  - Create two with content
  - Measure pixel distance difference

Plug-in Defense Details

- Tough to automate
- Includes page with top plug-in/add-ons
- Install and update manually
- Mozilla / Firefox 3: top 100
- Internet Explorer 8: top 100
- Safari: 48
- Opera: 26

Payload Defenses

- Scanning payloads can be neutralized
- File enumeration
  - BHO can hook on res:// ← sort of
  - Toughest thing to defend against presently
- Host scanning
  - smb:// alias all private IPs to 127.0.0.1
  - fiddler2 allows URL tampering separate from HOST
- Port scanning
  - XMLHTTPRequest foiled by aliases to google.com
  - Theoretical chance of leaking control
- Jikto: defending above solves this, too

Changing Fonts

- Windows keeps them in %windir%\fonts
- Requires a shell object to install or remove
- Removing requires removal of Registry key

Defense Tools

- Security toolbar or BHO
- Interception proxy and scripts
- x06d suite
  - sourceforge.net/project/x06d
  - JavaScript functions and BHO/Addon
  - Repository of client signatures
  - Performs POC for everything discussed so far

Future Defenses

- Scan non-text/html with clamav
  - Defeats bait and switch
  - Defeats simple trojan / phishing
- Duplicate with alternative browser
  - Diff the Response results, should be the same
  - Highlight in-line or sidebar
- Defang the page and scan again
  - Use the DOM: document.copy(TEMPFILE);
  - Redirect to TEMPFILE
  - Repeat until TEMPFILE does not change
  - Final Scan

Demos

- Phishing click-through
- Re-Anonymize, validate with panopticlick
- XSS click-through

Summary

- Browser defenses just getting reliable
- Tough to be thorough, but we should try
- Raising the noise level: herd defenses
- Framing another user: easier/better?
- Questions?