Analyzing JavaScript for Security Risks through Malware Ads

Chris Frisz

Brennon York

Thiago Rebello

Outline

- Introduction
- Identifying ads
  - Ad block survey
  - Analysis of ad blocker effectiveness
- Browser implementations
  - HtmlUnit
  - Firefox
- Data analysis
- Future work
- Conclusions

Introduction

- Goals
  - Develop a behavior-based algorithm to identify advertisements on web pages
  - Proactively block privacy-violating ads without the need for black lists.
- Steps taken
  - Developed ground truth for advertisements
  - Instrumented two web browsers

Presented by Thiao Rebello

ANALYZING ADS

Related Work

- Automatic Ad Blocking: Improving AdBlock for the Mozilla Platform
- An Effective Defense against Intrusive Web Advertising
- Phinding Phish: Evaluating Anti-Phishing Tools

Adblock Plus

- Perhaps the most famous ad blocker out in the web
- Blocks scripts, images, background, stylesheets, objects, xml-http requests, documents, element hiding
- Works through use of blacklists
- Lots of publicly available lists
  - EasyPrivacy and Fanboy's are most famous
  - Contain over 10,000 filters

Adblock Plus (Cont.)

- Users can choose what content to block
- Rules:
  - Wild cards (“*”) allow for matching before or after part of a URL or directory
  - “@@” define exception rules to prevent mismatches with certain filters
  - The “|” sign lets users choose where Adblock should match the expression, being in the beginning or end of the filter
  - Two “||” allow for matching against anything at the beginning of a domain name
  - Other rules include separator characters such as “^”

Ghostery

- Developed in a joint effort by online advertisement companies
- Also uses blacklists to block content
  - Content from ad providers is automatically blocked
  - No special rules
  - Users have no interaction and cannot choose to block or hide content
- List currently contains 504 subscribers

Ad Muncher

- Commercial software available for a fee
- Also contains a list of filters, but each filter contains one of the following set of actions (for example):
  - Remove links to URL
  - Remove all popups from URL
  - Remove divs/spans with text
  - Remove forms with text

Ad Muncher (Cont.)

- Provides browser interaction to allow users to block and hide content
- Intercepts winsock calls in memory and redirects traffic through itself
  - No browser configuration needed
  - Fast, easy and efficient

IE9 Tracking Protection

- Two ways to block content
  - Subscribe to lists
  - Automatic blocking
- Users cannot develop their own filters
- Users cannot choose content to block
- No implementation of regular expressions

IE9 Tracking Protection (Cont.)

- Rules:
  - The first line in each list contains a "msFilterList" header which denotes that the list is for tracking protection.
  - Each rule with a "–d" in front means that the rule blocks traffic from the domain. Similarly a "-" sign also means that traffic from the domain or directory is blocked.
  - Each rule with a "+d" in front means that requests to the domain are allowed.
  - When multiple lists target the same domain, the "Allow" rule wins and the URL is not blocked.

Ad Blocking Analysis

- Top 1,000 sites and bottom 1,000 sites from Alexa's top 1M sites
- HTTP Analyzer to gather URLs from sites
- Used personal profiles for Facebook, LinkedIn, and Orkut
- Used top 10 keywords from Google Insights from 2010 for search engines

Results

- Top 10 most common filters accounted for 31.9% of all ads found
- AdBlock Plus and IE9 filters blocked the exact same amount of ads with keyword ".com/ad?"
- Top 5 sites with the most ads matched accounted for 1.7% of all ads seen

Presented by Chris Frisz and Brennon York

BROWSER IMPLEMENTATIONS

Goals

- Examine primarily JavaScript-related functionality to identify advertisements
- Analyze:
  - JavaScript files and functions loaded
  - JavaScript functions executed
  - Function arguments
  - Modifications to the DOM tree

JavaScript

- How does it work?
  - JavaScript code is compiled to bytecode
    - Typically through a parser
  - Bytecode gets interpreted into machine code
    - Done through jump tables or a JVM
  - Native machine code can run on the local machine

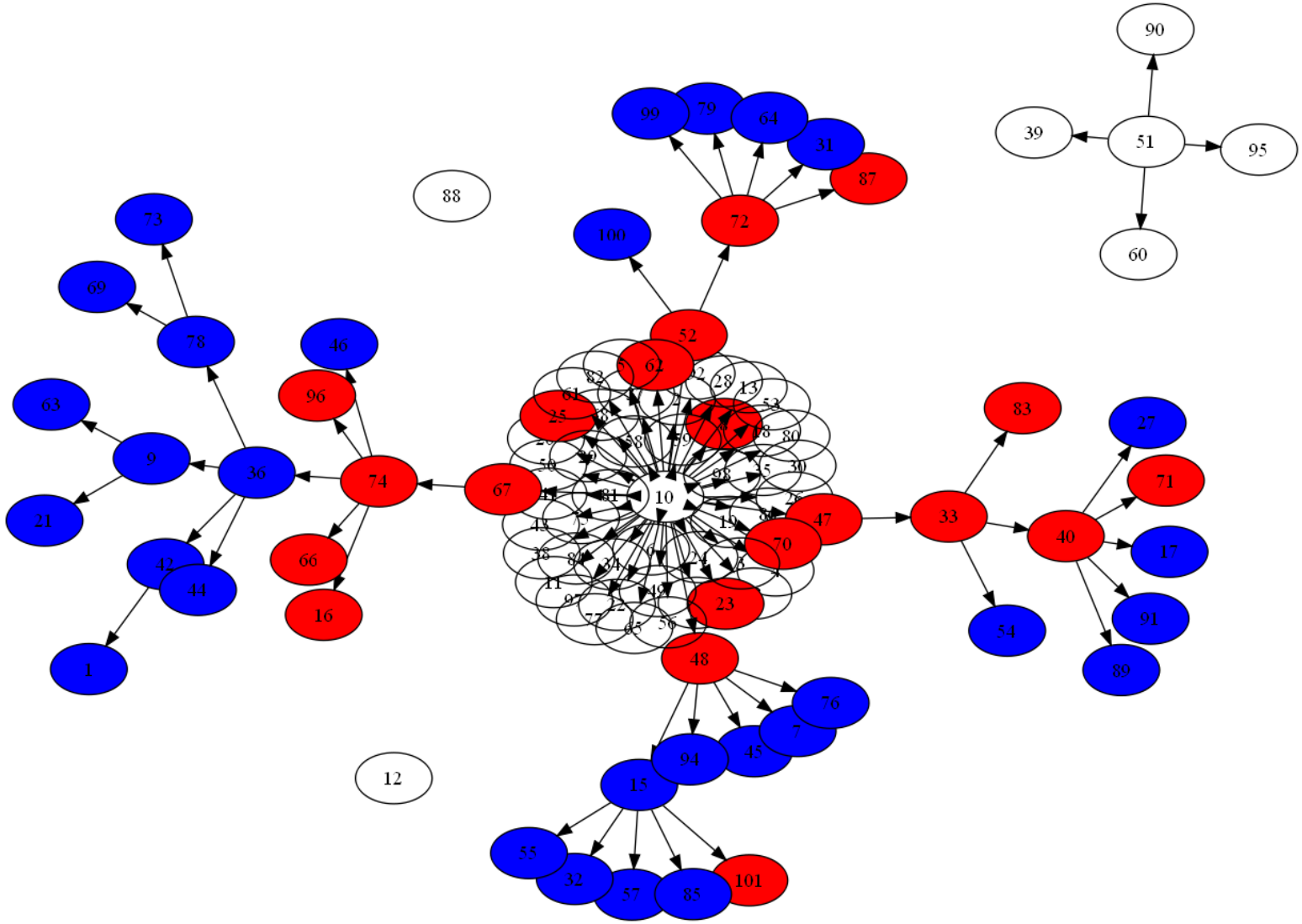
HtmlUnit

- Java-based
- Testing framework for web interactions
- GUI-less
- Utilizes Rhino JavaScript engine
  - A Java compiler
  - Pushes interpreted code to a locally running JVM

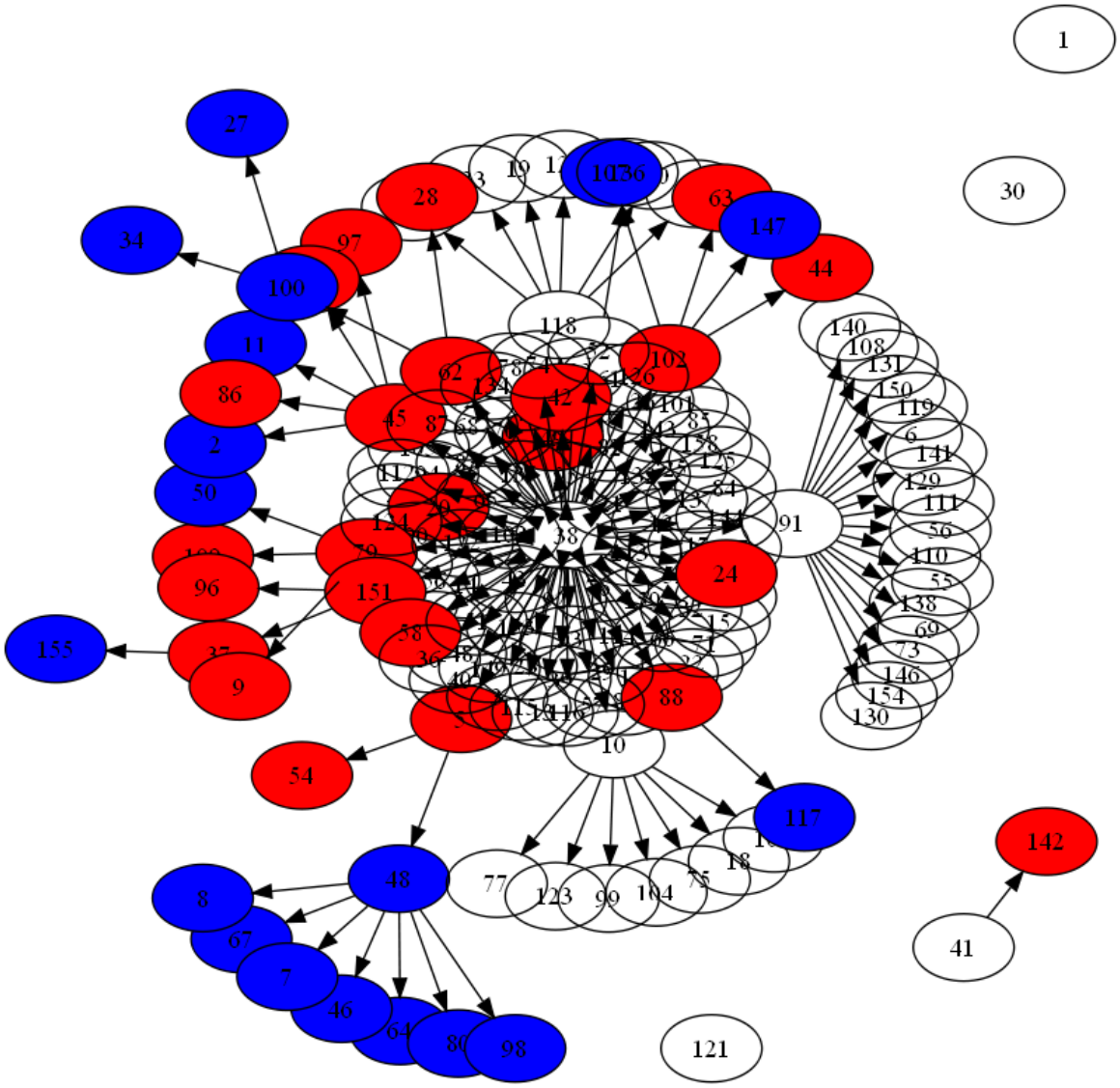
HtmlUnit (cont.)

- Recorded names of parsed JavaScript functions
- Instrumented to log all incoming URIs for a given web page
- Logged referrer fields from response headers

HtmlUnit (cont.)



HtmlUnit (cont.)



HtmlUnit Limitations

- Lost function invocations inside the JVM
- Obfuscation of image file formats
- Unable to visually verify accuracy of ad identification
- Prompted move to Firefox

Firefox

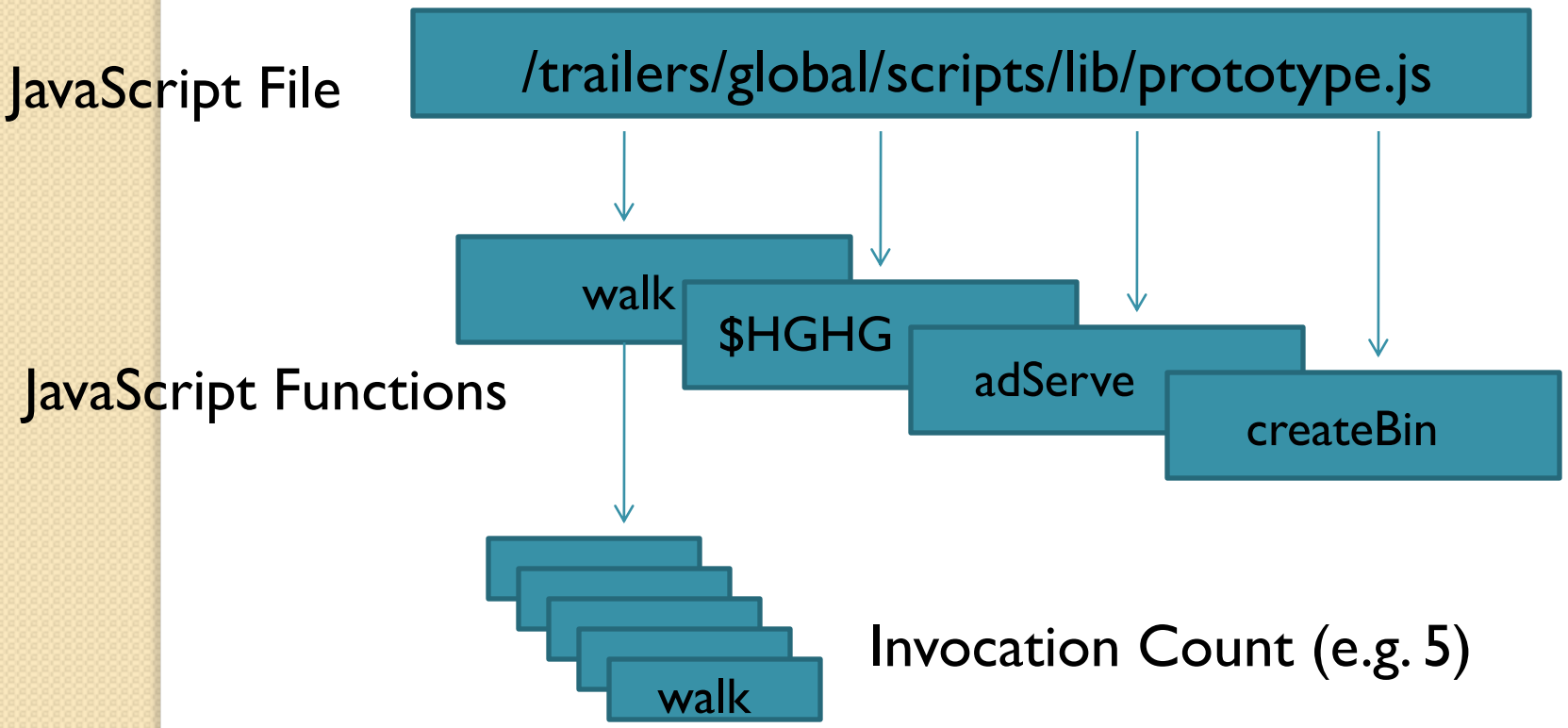
- Used source for version 4.0
- Included JägerMonkey JavaScript engine
  - Updated version of SpiderMonkey
- C/C++-based
  - Compiles code into a bytecode
    - Done in jsparse.cpp
  - Bytecode interpreted using jump tables to produce machine language
    - Done in jsinterp.cpp

Firefox (cont.)

- Re-implemented URI logging
- Re-implemented parsed function logging
- Added function invocation logging
  - Developed unique ID
    - Used to track between jsparse.cpp and jsinterp.cpp
- Used shared memory for log dumps

Data Collection

- Collected filename, function, and invocation count per function



Data Analysis

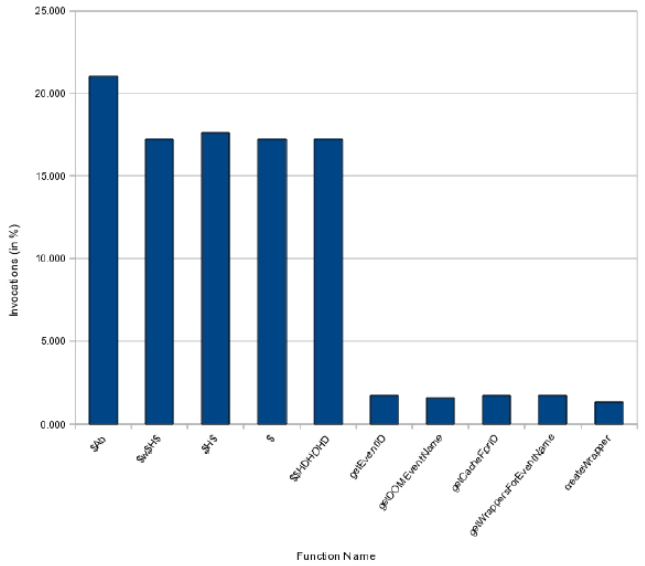
- Chose 10 sites with a wide range of advertisements
  - E.g. www.salon.com, www.cnn.com
- Compared those sites with the ground truth established previously to determine ad-related JS files

Data Analysis (Behavioral)

- For all JS files there exists a rapid decline in invocations
  - Typically 3-5 functions account for the vast majority of invocations
  - On average 31.1%

    of functions invoked

    on load



Data Analysis (Behavioral)

- Average of ~24 JS files loaded for each web page observed
- Follows two forms:
  - Large number of small files loaded
    - Typical invocation pattern
  - Single file with large number of functions

Future Work

- Include client-side interaction into data
- Examine changes in the DOM tree
- Analyze response headers from URIs
- Develop methodology for identifying obfuscated function names
- Create browser-based tool/plug-in to proactively block advertisements

QUESTIONS/COMMENTS?

References

[1] Justin Crites and Mathias Ricken. Automatic Ad Blocking: Improving AdBlock for the Mozilla Platform. Rice University. Houston, TX.

[2] Writing Adblock Plus Filters. Adblockplus.org. Adblock Plus, n.d. Web. 2 May 2011.

[3] Element Hiding Helper. Adblockplus.org. Adblock Plus, n.d. Web. 2 May 2011.

[4] An Effective Defense against Intrusive Web Advertising. Viktor Krammer, Secure Business Austria, Vienna University of Technology, A-1040 Vienna, Austria

[5] NoScript. InformAction., n.d. NoScript JavaScript/Java/Flash blocker for a safer Firefox Experience! what is it? InformAction. Web. 2 May 2011.

[6] Privoxy. Privoxy Developers, n.d. Privoxy Home Page. Web. 2 May 2011.

[7] Ad Muncher Ad Muncher Wiki. Admuncher.com. Ad Muncher, n.d. Web. 2 May 2011.

[8] FAQ Adblock Plus internals. Adblockplus.org. Adblock Plus, n.d. Web. 2 May 2011.

[9] IEInspector HTTP Analyzer HTTP Sniffer, HTTP Monitor, HTTP Trace, HTTP Debug. Ieinspector.com. IEInspector Software LLC, n.d. Web 2 May 2011.

[10] Ghostery. Evidon, Inc., n.d. Web. 2 May 2011.

[11] Privacy protection and IE9: who can you trust? Zdnet.com. Ed Bott, ZDNet, 14 Feb. 2011. Web. 2 May 2011

[12] IE9 follows Firefox 4s lead on Do Not Track Computerworld. Computerworld.com. Gregg Keizer, Computerworld, 16 Mar. 2011. Web. 2 May 2011

[13] Zhang, Yue, Serge Egelman, Lorrie Cranor, and Jason Hong. "Phinding Phish: Evaluating

Anti-Phishing Tools." Carnegie Mellon University. Pittsburg, PA.

[14] Google Insights for Search. Google.com. Google, n.d. Web. 2 May 2011.

[15] Cova, Marco, Christopher Kruegel, and Giovanni Bigna. "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code." Raleigh, North Carolina, 2010.

[16] Curtsinger, Charles, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. "Zozzle: Low-overhead Mostly Static JavaScript Malware Detection." Technical Report, Microsoft Research, 2010.

[17] Jang, Dongseok, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. "An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications." Chicago, Illinois, 2010.

[18] Meyerovich, Leo and Benjamin Livshits. "ConScript: Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser." Oakland, California, 2010. 11