Javascript Static Security Analysis made easy with

Nishant Das Patnaik  
@dpnishant

Sarathi Sahoo  
@sarathisahoo
Agenda

• Introduction to the problem
  - Why is it a problem?
  - What is the impact?
  - Demo

• What is JSPrime?
  - What is it?
  - Who is it for?
  - How it works?
  - What it can do?
  - What it can’t do?
  - Demo

• Conclusion and questions
Who am I?

• First time BlackHat speaker
• Senior Paranoid at Yahoo! Inc.
  • Security Engineer at eBay Inc. (Past)
• Bug Bounty Hunter
• Speaker at NullCon 2012, Goa, India
• Co-author of Ra.2: – DOM XSS Scanner Firefox add-on
• 5+ years of security self-studying
• Keyboard Player & Sports-bike enthusiast
Who is Sarathi?

- Experienced Application Developer, 7+ years experience
- 5+ years at Yahoo! Inc.
- Full-time JSPrime Developer
- @sarathisahoo, http://fb.me/sarathi.sahoo
JavaScript: the lingua franca of Web & Mobile
Introduction: The Problem

JavaScript is a *dynamic* language

- Object-based, properties created on demand
- Prototype-based inheritance
- First-class functions, closures
- Runtime types, coercions
Introduction: The Problem

- Client Side Script Injection
  - DOM XSS

- Server Side Script Injection
  - Node.JS Applications
Introduction: Why is it problem?

- Server side filtering fails for DOM XSS
- JavaScript code review is intimidating #iykwim
- Library dependent source-to-sink pairs
- Not Enough Scanners
Introduction: The Impact

• Same as *regular* XSS: Reflected or Stored

• Script Injection on server side or mobile device can be really lethal.

• Node.JS, Firefox OS, Windows 8 Apps (WinJS)
Vulnerability Demo

Some videos or sample codes
Introducing JSPrime

- What is it?
- Who is it for?
- What it can do? Avoiding False positives
- What it can’t do? Knowing the False negatives
- Stability & Automation
- Demo
Introducing JSPrime: What is it?

• JSPrime is a light-weight source code scanner for identifying security issues using static analysis.

• It is written in Javascript to analyze JavaScript.

• Uses the open-source ECMAScript parser: Esprima.org
Introducing JSPrime: Who is it for?

- JSPrime is mostly a developer centric tool.
- It can aid code reviewers for identifying security issues in 1st pass.
- Security professionals may find it useful during penetration testing engagements.
Introducing JSPrime: How it works?

• Feed the code to Esprima, to generate the AST.
• Parse the JSON AST, to locate all sources (including Objects, Prototype) and keeping track of their scopes
• Parse the AST, to locate all assignment operations related to the sources, while keeping track of their scopes
• Parse the AST to locate sinks and sink aliases, again keeping track of their scope.
• Parse AST to locate functions (including closures, anon functions) which are fed with sources as arguments and while tracking down their return values.
Introducing JSPrime: How it works?

- Once all the sources, source aliases are collected we check for any filter function on them, rejected if found.
- Remaining sources, source aliases are tracked for assignments or pass as argument operations to the collected sinks or sink aliases.
- We repeat the same process in reverse order to be sure that we reach the same source when we traverse backwards, just to be sure.
- Once we confirm that we extract the line numbers and their statement and put it in the report we generate with different color coding
Introducing JSPrime: What it can do?

• It can follow code execution order
• Handle First-class functions
• Analyze Prototype-based inheritance
• Understand type-casting
• Understand context-based filter functions (has to be manually supplied, though)
• Library aware sources and sinks
• Variable, Objects, Functions scope aware analysis
• Control-flow analysis
• Data-flow analysis
Introducing JSPrime: What it can’t do?

- It can’t detect 100% of the issues.
- It can’t learn sources and sinks automatically.
- It can’t handle obfuscated JavaScript.
- It can’t report issues in minified JavaScript, unless beautified.
- It can’t analyze dynamically generated JavaScript using ‘eval’ or similar methods.
Introducing JSPrime: Stability & Automation

• Handle up to 1500 LoC in a single scan
• Node.JS port is available for server-side web service like setup
• Largely dependent on Esprima’s robustness, can be the 1st point failure
Demo

Have patience! 😊
Roadmap

- Improved performance and stability
- Multiple file scanning
- Node.JS Project Scanning capability
- IDE Plugin (Notepad++, WebStorm, ??)
- More Library Support
- String manipulation simulation

Your suggestions? 😊
Summary

Actively work-in-progress
Promising project roadmap
Open-sourced today

www.jsprime.org
Credits

• Aria Hidayat, Esprima.org
• Paul Theriault, Mozilla Security Team
• Bishan Singh - @b1shan
• Rafay Baloch – rafayhackingarticles.com
Questions?
THANK YOU