



# Detecting Bugs at Scale with Semgrep

Parsia Hakimian  
[parsiya.net](http://parsiya.net)

# TL;DW

Too Long;  
Didn't Watch!



*Image credit: Myself - The Semgrep Logo belongs to r2c*

# Yours Truly

2013-2019: Digital/Synopsys - consulting

Web -> Mobile -> Thickclients/Blockchain (lol)

DEF CON 26

Tineola Taking a Bite Out of Enterprise Blockchain

2019 - 2022: Electronic Arts - security “engineer”

Videogames -> Dynamic testing ->

Source code review -> Semgrep

DEF CON 28 Appsec Village

localghost: Escaping the Browser Sandbox without 0-Days

2022: Unemployed and a burden on society



# Episode 1: Static Analysis

Image credit: Lost in Random, Zoink/Electronic Arts



# Inside You There are Two Wolves

Security Engineer

Developer



Image credit: Victor Becker  
Lost in Random, Zoinik/Electronic Arts

# Inside Your SAST There are Two Wolves

<b>For Security Engineers</b>	<b>For Developers</b>
Some noise is OK.	Any noise is bad!
Can be slow.	Must be quick*.
Manual usage is tolerated.	Should be part of CI/CD.
Alert on everything.	Never break the build.
Configuration/tuning is expected.	The tool should work out of the box.

# Episode 2: Semgrep



Image credit: Lost in Random, Zoink/Electronic Arts  
The Semgrep logo belongs to r2c

# Semgrep vs. CodeQL vs. Coverity vs. ... !



## Mysterious Playing Cards

These playing cards are beautifully painted and, in some angles, they seem to glow a faint blue...

[E](#) Continue

Image credit: Lost in Random, Zoink/Electronic Arts

# Rules are Easy!

```
rules:  
  - id: arrays-passed-to-functions  
    patterns:  
      - pattern-either:  
        - pattern-inside: |  
          $TYPE $BUF[$SIZE] = $EXPR;  
          ...  
        - pattern-inside: |  
          $TYPE $BUF[$SIZE];  
          ...  
        - pattern-inside: |  
          $TYPE $BUF[] = $EXPR;  
          ...  
      - pattern: $FUNC(..., $BUF, ...);
```

```
1 int main() {  
2   int oneDim[10];  
3   int multipleDims[10][20][30][40] = {0};  
4  
5   // ruleid: arrays-passed-to-functions  
6   int ret = func(oneDim);  
7  
8   // ruleid: arrays-passed-to-functions  
9   int ret2 = func(30, multipleDims, 20, 40);  
10  
11  
12  
13  
14  
15
```

# Semgrep Limitations

A screenshot from the video game "Lost in Random". A young girl with long, flowing orange hair stands in a dimly lit, green-hued environment, looking towards a large, dark, skeletal creature. The creature has a pale, glowing face and long, thin limbs. To the left, a large, white, porous cube-like creature with a single eye and a small mouth is visible. The background is filled with hanging flags and a sense of mystery.

Weak multi-file analysis - Limited taint tracking  
Join Mode, DeepSemgrep (paid)

Limited set of available rules  
We must create custom rules

# Episode 3: The Lay of the Land



Image credit: Leo Brynielsson  
*Lost in Random*, Zoink/Electronic Arts

# Code, Configs and more Code!



Source Code

Image credit: Lost in Random, Zoink/Electronic Arts



# Dependency Hell

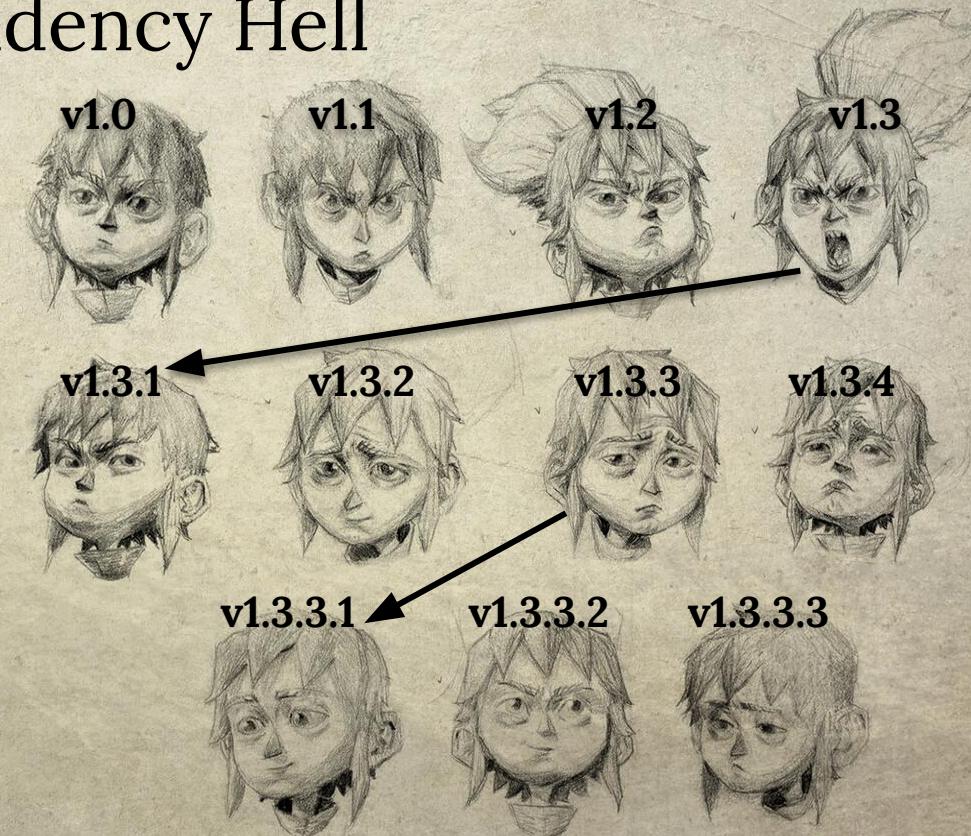


Image credit: Victor Becker  
*Lost in Random*, Zoink/Electronic Arts

EA Originals Zoink!

Skip E

# Source Code Management

What in the world?!

Perforce

Image credit: Lost in Random, Zoink/Electronic Arts

# Webview Chromium Desktop Apps

Electron



Edge Webview2



CEF

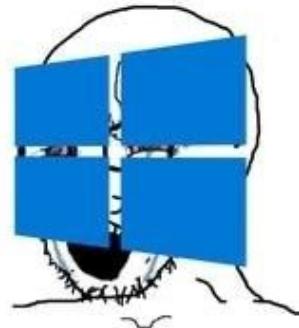


QtWebEngine





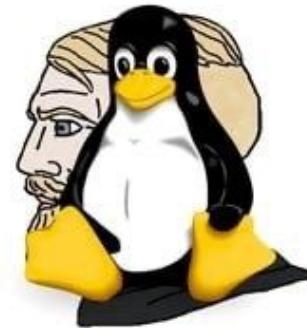
hey can i uninstall edge



NOOO!!! YOUR SYSTEM WILL  
BREAK



im going to uninstall the  
bootloader



go ahead lol

# What to Look For?

Normal webapp client-side and server-side tests.

XSS -> RCE is a meme but mostly true.

As easy as

<img src=x

onerror=document.location="file:///C:/windows/system32/calc.exe">

Special Webview App Things:

Protocol Handlers. E.g., slack://blah.

Local servers. E.g., [RCE in PlayStation Now](#) and [RCE in VS Code WSL Extension](#).

[The JavaScript Bridge](#) or what can you call after an XSS?

Client-side Storage

# Episode 4: Automation



Now roll the dice...

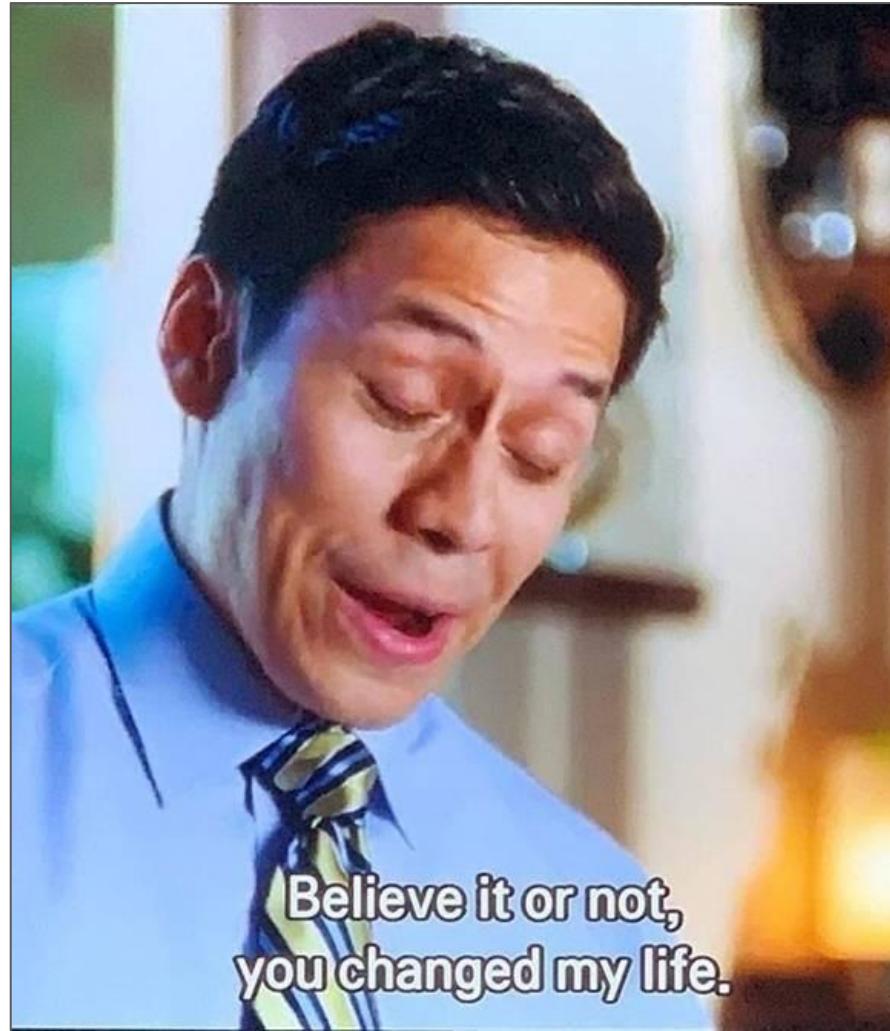
Image credit: Lost in Random, Zoink/Electronic Arts

# Manual Work is a Bug

A.B.A: always be automating

1. Documenting the steps.
2. Create automation equivalents.
3. Create automation.
4. Self-service and autonomous systems.

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



# Literature Review

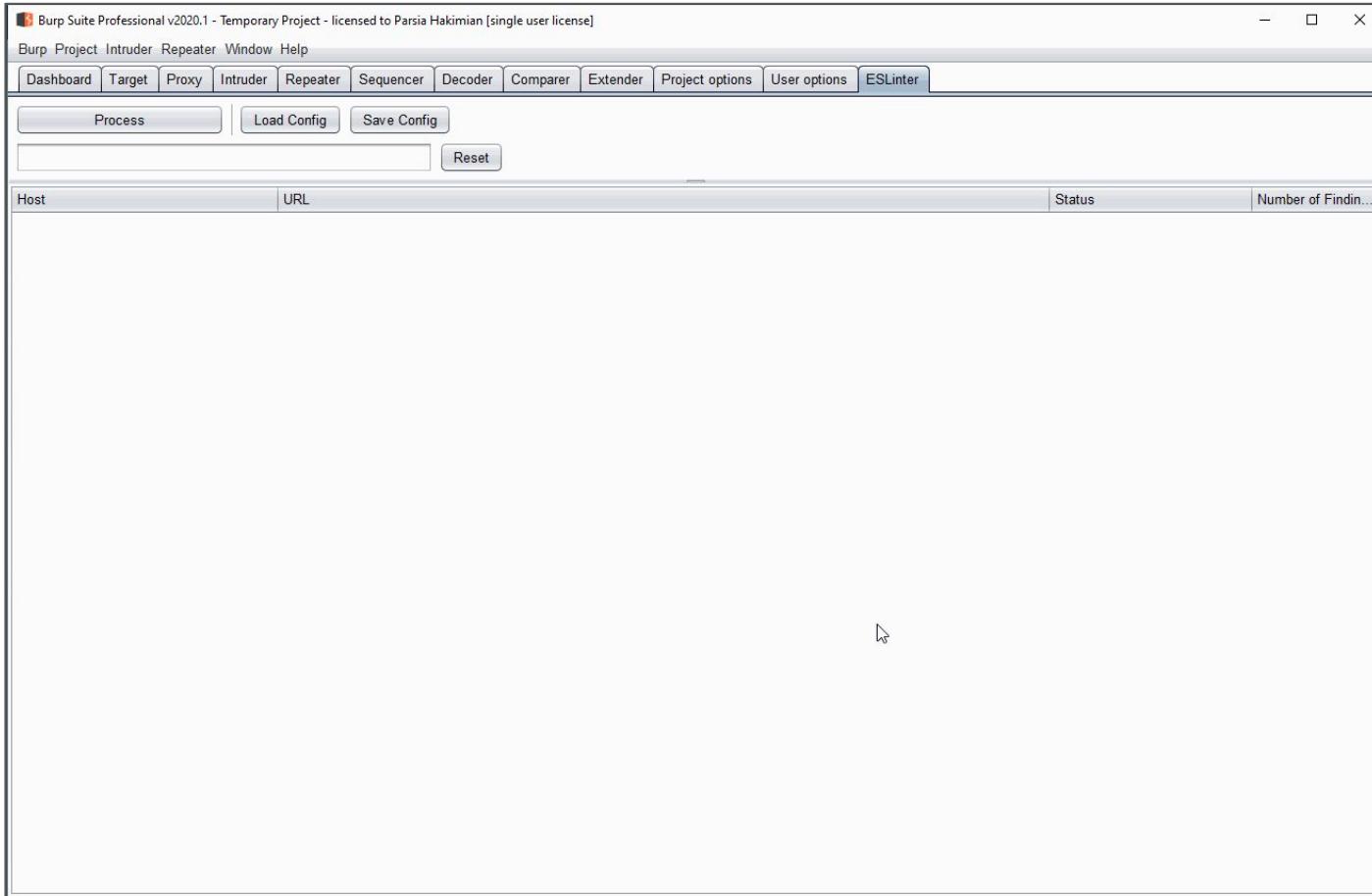
A dark, atmospheric scene from the movie Frozen 2. Elsa, dressed in her signature black and white ice-themed gown, stands on the right side of the frame, holding a glowing purple orb. She is positioned in front of a large, jagged ice structure with several tall, thin spires of ice reaching upwards. The background is filled with more ice formations and a hazy, purple-tinted light filtering through the cracks in the ceiling. In the foreground, there are smaller, dark, silhouetted figures that appear to be ice statues or perhaps other characters in shadow.

Academic Papers

Industry

Netflix (lol)

# Some Early Automation: ESLinter



# Semgrep Usage

What rules are used?

Where does the output appear?

How is Semgrep executed?

# For Security Engineers - How?

How do you run it?

It can be manual.

```
$ semgrep --config /myrules/ --sarif --output myresults.sarif /path/to/code
```

## Quick Start

1. Clone the repository.
2. Run Semgrep with `--config` pointing to the repository or any of the subdirectories.

```
$ semgrep --config /path/to/semgrep-hotspots/ . --sarif --output my-results.sarif
```

# For Security Engineers - Output

Where is the output stored?

Use a SARIF viewer.

The screenshot shows a developer's workstation with two main windows. On the left is a code editor displaying `StringUtils.java`. The code includes a warning about SHA1 collisions and a call to `MessageDigest.getInstance("SHA-1").digest(data)`. On the right is a SARIF viewer titled "18 SARIF Results". It shows a tree view of findings across multiple files, with a detailed view of the SHA1 collision warning from `StringUtils.java`.

**SARIF Results Tree:**

- > SystemUtils.java utils 8
- > Database.java database 2
- < StringUtils.java utils 2
  - ⚠ 98 SHA1 Hash algorithm used. The SHA1 hash is known to have hash collisions.
- > BurpTab.java gui 1
- > Beautify.java lint 1
- > Detective.java burp 1
- > Metadata.java lint 1
- > ProcessRequestTask.java lint 1
- > ReqResp.java utils 1

**LOCATIONS 9 RULES 7 LOGS 1**

**Filter results**

# For Security Engineers - Rules

Array Access

snprintf

Uninitialized  
Pointer

Image credit: Lost in Random, Zoink/Electronic Arts

# Detecting XXE in Java

Let's find every Java XML parser!

## SEMGREP RULE

Simple Advanced

```
rules:
- id: blog-2022-03-java-xxe
  patterns:
    - pattern-either:
      - pattern: import DocumentBuilderFactory;
      - pattern: import XMLInputFactory;
      - pattern: import DOMParser;
      - pattern: import TransformerFactory;
      - pattern: import Validator;
      - pattern: import SchemaFactory;
```

## TEST CODE

```
● 1  import javax.xml.parsers.DocumentBuilderFactory;
● 2  import javax.xml.stream.XMLInputFactory;
● 3  import oracle.xml.parser.v2.DOMParser;
● 4  import javax.xml.transform.TransformerFactory;
● 5  import javax.xml.validation.Validator;
● 6  import javax.xml.validation.SchemaFactory;
● 7  import javax.xml.transform.sax.SAXTransformerFactory;
● 8  import org.xml.sax.XMLReader;
● 9  import org.dom4j.io.SAXReader;
● 10 import org.jdom2.input.SAXBuilder;
● 11 import javax.xml.bind.Unmarshaller;
● 12 import javax.xml.xpath.XPathExpression;
● 13 import java.beans.XMLDecoder;
14
15 import javax.whatever.Whatever;
```

# Keywords in Function Names - Encode/Decode

## encode-decode-in-function-name ⓘ

simple mode advanced mode

```
1 rules:
2   - id: encode-decode-in-function-name
3     patterns:
4       - pattern: |
5         $FUNC(...);
6       - metavariable-regex:
7         metavariable: $FUNC
8         regex: .*(encode|decode).*
9       message: Encoding/decoding function, worth a review.
10      languages:
11        - cpp
12      severity: WARNING
13
```

test code metadata docs

```
1 int main() {
2
3   // ok:encode-decode-in-function-name
4   randomfunc();
5
6   // ruleid:encode-decode-in-function-name
7   encodeme();
8
9   int param = 1;
10  // ruleid:encode-decode-in-function-name
11  int res = decodeme(param);
12
13  // ruleid:encode-decode-in-function-name
14  thisencodefunctionisbad();
15
16  // ok:encode-decode-in-function-name
17  int res2 = safefunction();
18
19
20 }
```

# And there's more!

Other tickets - incoming vulnerability reports

Find the root cause in source code and write rules.

Crypto(graphic) primitives - MD5

MD5

Memory-safe language - Unsafe

Hardcoded secrets

C++ - <https://github.com/parsiya/semgrep-hotspots>

sizeof(pointer\*)

Snprintf

# For Developers

My manager

Me

Dev team looking for  
security reviews

Child! COVER US!!

Image credit: Lost in Random, Zoink/Electronic Arts

# For Developers - Output

Parse the SARIF results and do whatever.

<https://github.com/microsoft/sarif-tools>

<https://github.com/parsiya/sarif-info>

We explored some options like

Automatic tickets in Jira or ... .

Slack notifications.

Pull/Merge request comments.

# For Developers - Rules

Ask them!

What keep you up at night?

Which section of the codebase is vulnerable?

What kind of security issues are important to you?

Show them!

How rules are written.

How Semgrep works.

They will start writing their own rules.

# For Developers - Rules

Don't have noise! You will lose their trust!

test code metadata docs

```
1 int main() {  
2  
3     // ok: out-of-bounds-array-access  
4     int buf[100];  
5     buf[10] = 10;  
6  
7     // ruleid: out-of-bounds-array-access  
8     int buf2[100];  
9     buf2[100] = 100;  
10  
11    // ok: out-of-bounds-array-access  
12    int buf3[100];  
13    int nem = buf3[10];  
14  
15    // ruleid: out-of-bounds-array-access  
16    int buf4[100];  
17    int nem = buf4[120];  
18 }
```

Matches

↑Line 8

Out of bounds array access for buf2 in 'buf2[100]' (100 >= 100).

↑Line 16

Out of bounds array access for buf4 in 'buf4[120]' (120 >= 100).

# Don't Break the Build!

Developer

You



# Moral of the Story



Security Engineer

Developer

Actual Bugs

# What Did We Learn Here Today?

Semgrep is free and fun. Please use it!

Write your own rules.

Keep the noisy rules for yourself.

Send accurate rules to the developers.

Try to make the process as seamless as possible for them.

Encourage the developers to experiment with Semgrep and write rules.

# Bonus: Where Do I start?

Static Program Analysis: <https://cs.au.dk/~amoeller/spa/>

Learn Semgrep: <https://semgrep.dev/learn>

Protocol Handlers:

<https://parsiya.net/blog/2021-03-17-attack-surface-analysis-part-2-custom-protocol-handlers/>

Local Servers:

[localhost: Escaping the Browser Sandbox without 0-Days](#)

[RCE in PlayStation Now](#)

[RCE in VS Code WSL Extension](#)

[The JavaScript Bridge in Modern Desktop Applications](#)

# Bonus: Where Do I start? Continued

Manual Work is a Bug

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

ESLinter: Manual JavaScript Linting is a Bug

<https://github.com/parsiya/eslinter>

Semgrep in the CI/CD Pipeline

[GitHub Action](#)

[GitLab SAST](#)

[OWASP secureCodeBox](#)

Semgrep Hot Spots

<https://parsiya.net/blog/semgrep-hotspot/>

<https://github.com/parsiya/semgrep-hotspots>

# Bonus: Where Do I start? Continued

Semgrep Deployment

[Building a SAST program at Razorpay's scale](#)

[A Guide On Implementing An Effective SAST Workflow](#)

# Questions

[twitter.com/CryptoGangsta](https://twitter.com/CryptoGangsta)

[parsiya.net](http://parsiya.net)

[github.com/parsiya](https://github.com/parsiya)



Image credit: Lost in Random, Zoink/Electronic Arts

 EA Originals Zoink!