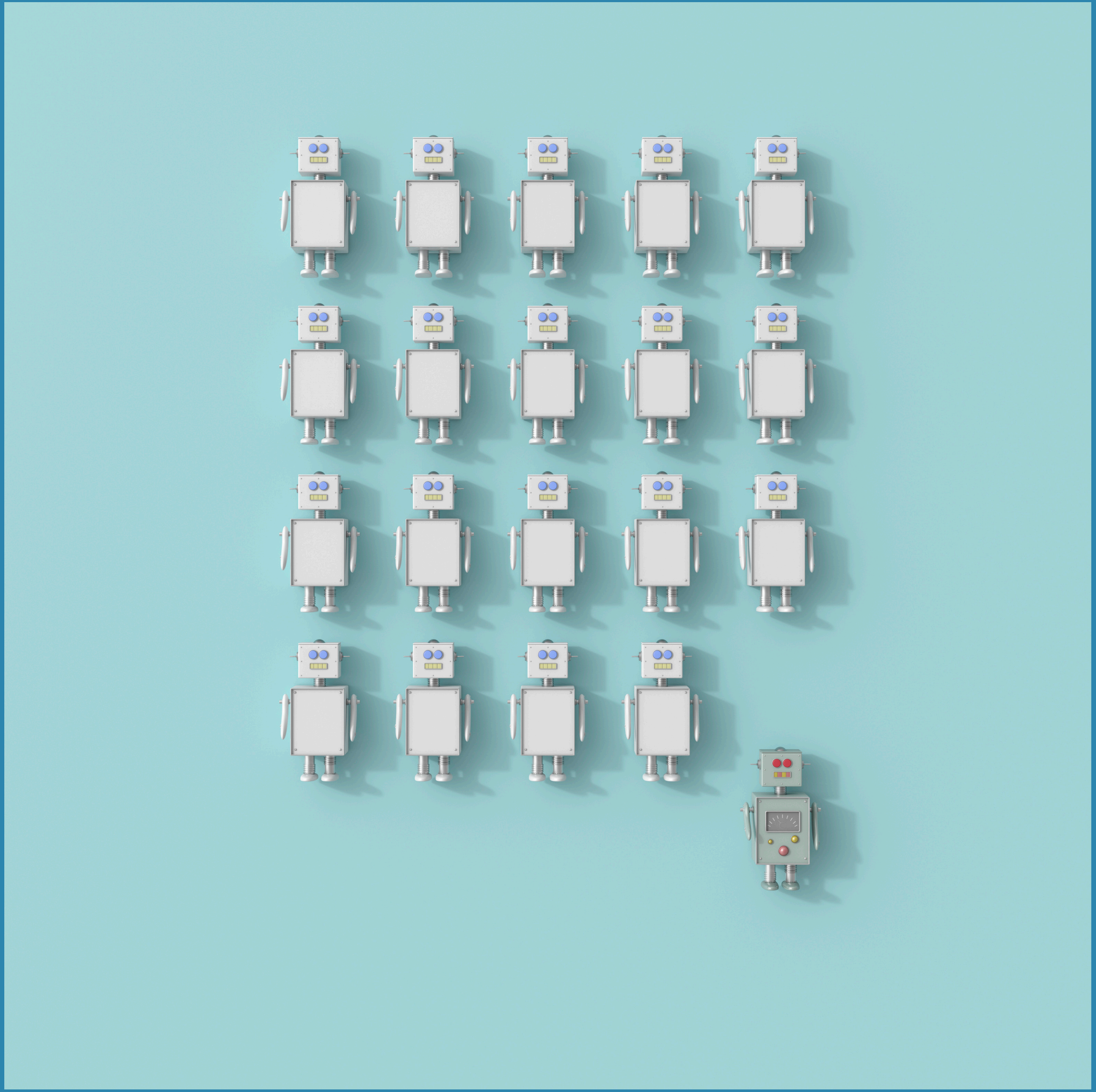


Automation for Manual Bug Bounty Hunters

Eugene Lim

@spaceraccoon

Eugene Lim 🇸🇬
spaceraccoon

Blog: spaceraccoon.dev

Twitter: @spaceraccoonsec

Why automate?

My bug bounty journey mostly involved manual hunting.

Jan 2019

- IDORs, business logic, XSS – First bounty end-Feb 2019 IDOR
- Nov 2019: H1-213 Live Hacking Event Most Valuable Hacker
- Large number of programs and bugs
- Entered global all-time top 100 by Mar 2020

2022

- Native vulnerabilities, reverse engineering, code review
- Jan 2021: H1-Elite Hall of Fame
- Selective programs and higher impact
- Still global top 40, but not really farming rep

In the long run, bounty hunters tend to fall into two camps.

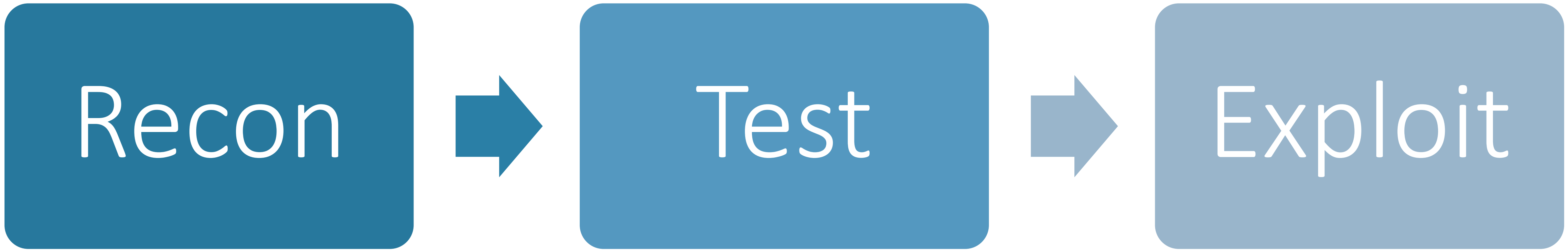
Automation	Manual
• Recon	• Scope
• Services	• Applications
• Scanners	• Humans
• Signatures	• Heuristics

Despite overlaps, they require different skillsets and resources.

- Subdomain enumeration
- Attack surface mapping

- Signature-based scanners
- Fixed payload templates

- Automated exploitation

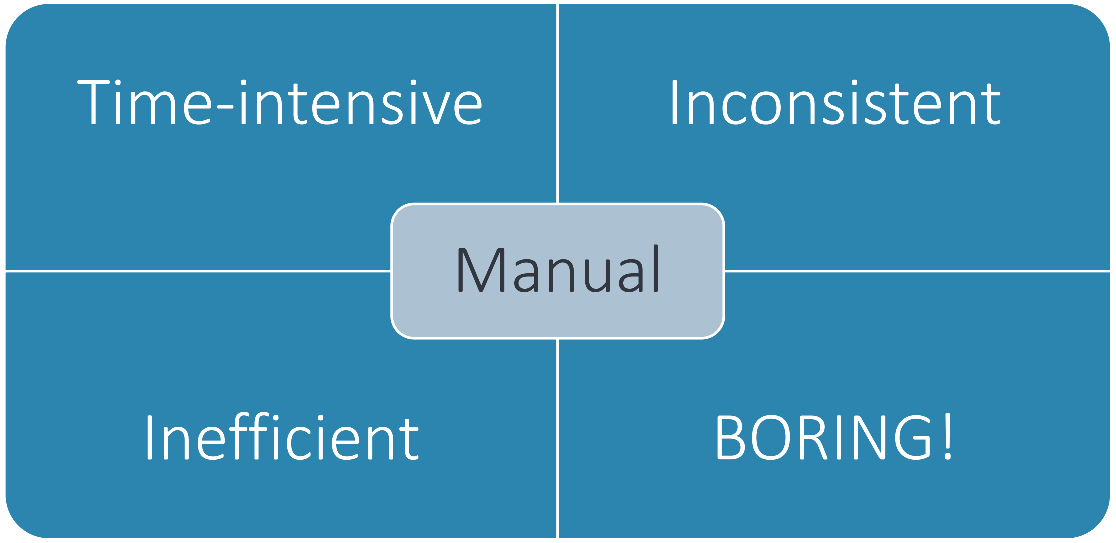


- Explore application functionality
- Enumerate APIs
- Decompile source code

- Manually modify payloads
- Observe behavior

- Custom escalation and pivoting

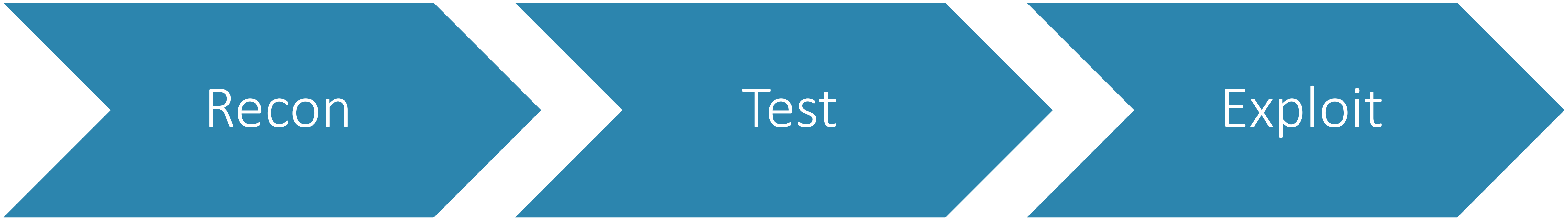
However, manual hunters should not rely exclusively on “browser-and-Burp”.



Free tools have increased in quantity and quality…

Fuzzing	Mobile	SAST
clusterfuzz	AppShark	Semgrep
RESTler	MobSF	CodeQL

…helping manual hunters focus on triage and exploitation.



Recon

- Automate attack surface mapping
- Manually define search sources

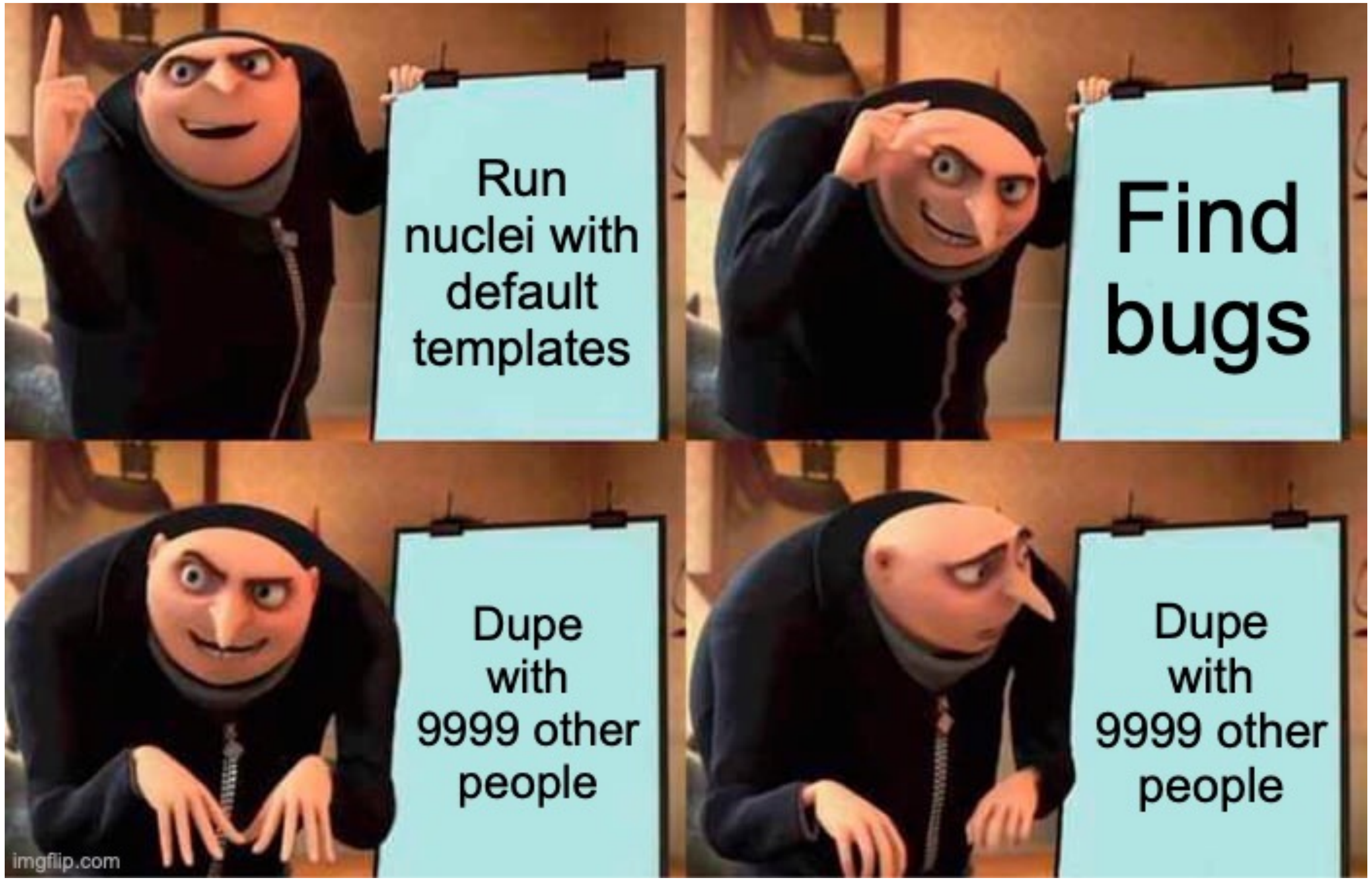
Test

- Filter interesting behaviour
- Manually confirm bug

Exploit

- Automate payload generation
- Manually write generators

More importantly, they help manual hunters differentiate themselves from automation and recon hunters.

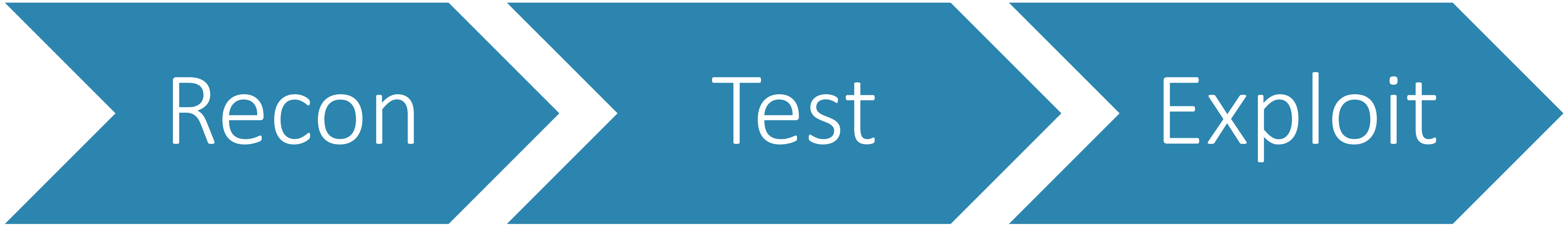

Run nuclei with default templates
Find bugs
Dupe with 9999 other people
Dupe with 9999 other people
imgflip.com

Automation for manual hunters uses a different set of tools and workflows from standard automation.

“Rather than scanning for vulnerabilities, we need to scan for *interesting behaviour*. Then, having identified the tiny fraction of inputs that yield interesting behaviour, we can investigate further.”

– James Kettle, “Backslash Powered Scanning: hunting unknown vulnerability classes”

Today, I will discuss how you can add automation in each stage as a manual bug bounty hunter.



Recon

Single-app scope doesn't have to mean zero recon.

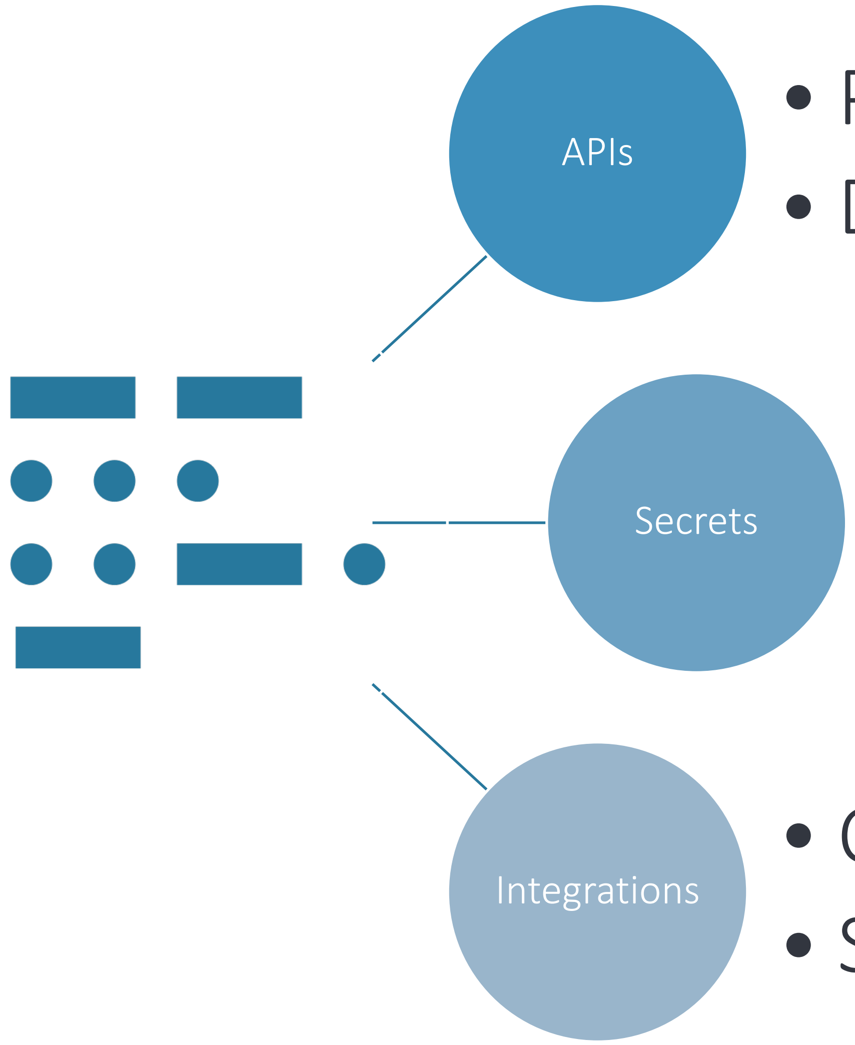
Client-side source code

API enumeration / introspection

Cloud / SaaS integration

Reverse engineering

Client-side code can yield important data.



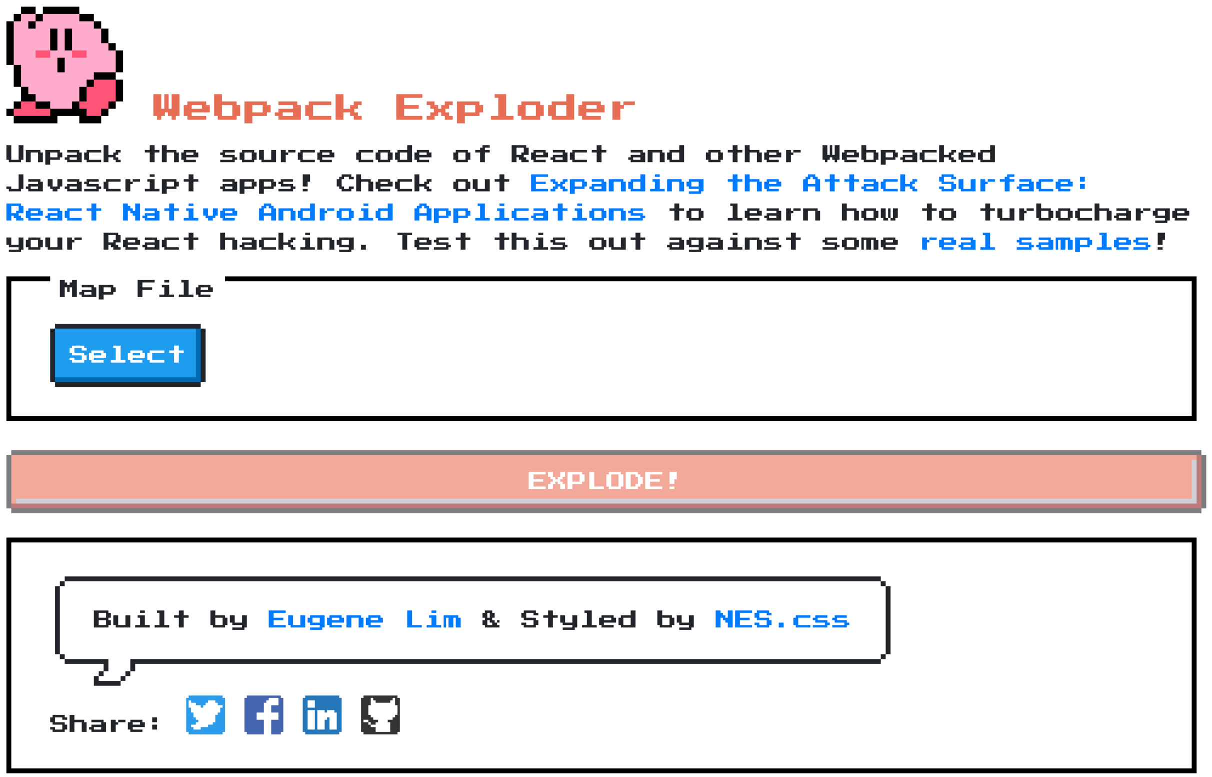
- Prod
- Dev

- Credentials
- Encryption

- Cloud
- SaaS

1-to-1 (or close to) decompilers are key!

- webpack-exploder
- asar
- wabt
- hbctool



Unpack the source code of React and other Webpacked Javascript apps! Check out Expanding the Attack Surface: React Native Android Applications to learn how to turbocharge your React hacking. Test this out against some real samples!

Shoutout to LinkFinder for quick API extraction

Common API frameworks and how to enumerate them.

GraphQL	Introspection*
OpenAPI	Swagger UI
Apollo	Suggestions + Playground
Generic	Dirbusting + Documentation

Keep an eye out for third-party integrations.

Cloud

- Resource enumeration (letitgo, recon.cloud)
- Bucket naming patterns
- Role permissions

SaaS

- Dangling urls
- Shortlink services
- CORS/PostMessage misconfigurations

Reverse-engineering (dynamic and static) is a highly-underrated skill in bug bounty.

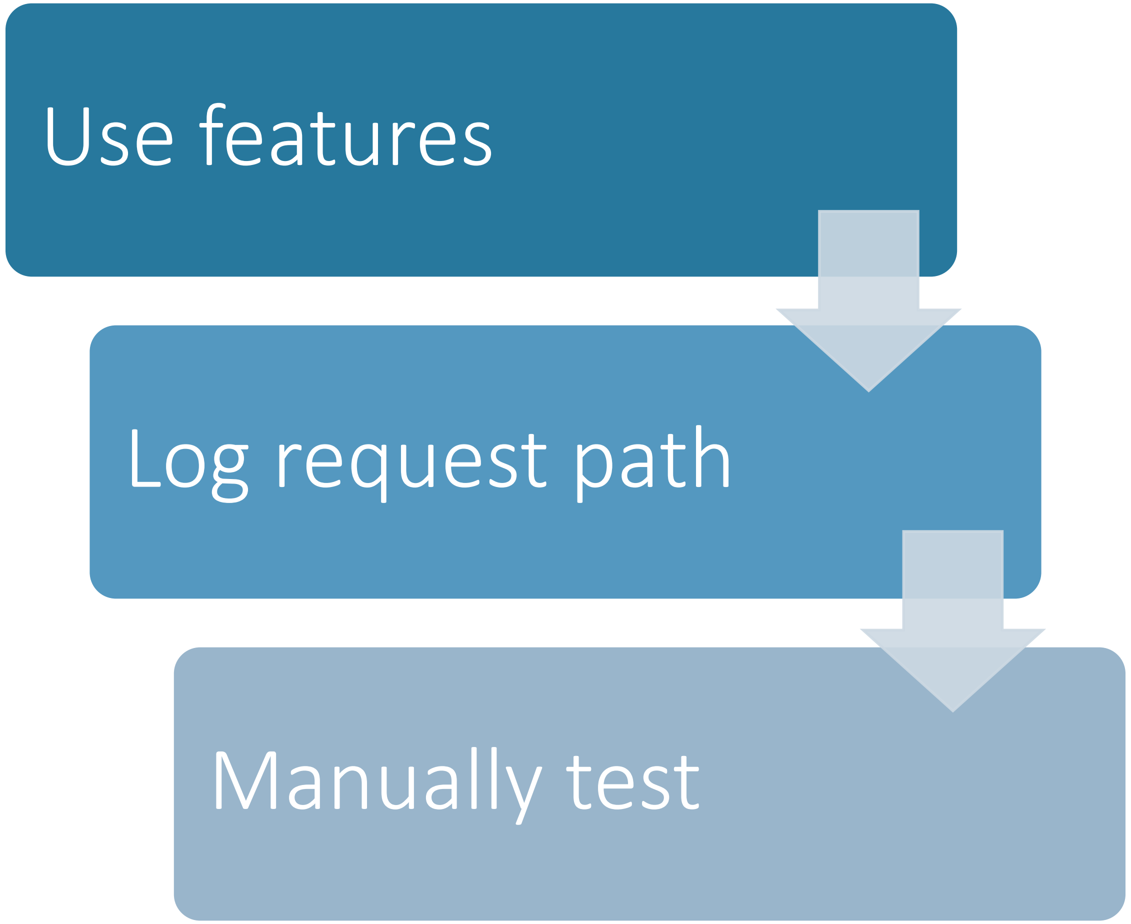
Frida

Radare

Ghidra

Test

My old spreadsheet method…



Path	Method	Params	XSS	SSRF
/users	GET	id	X	X
/users	POST	email	X	X
/users	PUT	name	Y	X
…	…	…	…	…

As I moved to part-time hunting, this became untenable.

Positives

- Extremely thorough
- Customized for each context
- Easy to bootstrap

Negatives

- Extremely slow
- Inconsistent and not codified
- Difficult to scale

Leverage modern DevSecOps tools to pre-screen possible vulnerabilities.

Static	Dynamic
• Semgrep	• ClusterFuzzLite
• CodeQL	• Jaeles
• AppShark*	• RESTler

Semgrep / CodeQL quickly identifies potential vulnerabilities in huge codebases.

Automatically identify vulnerable **sources and sinks**

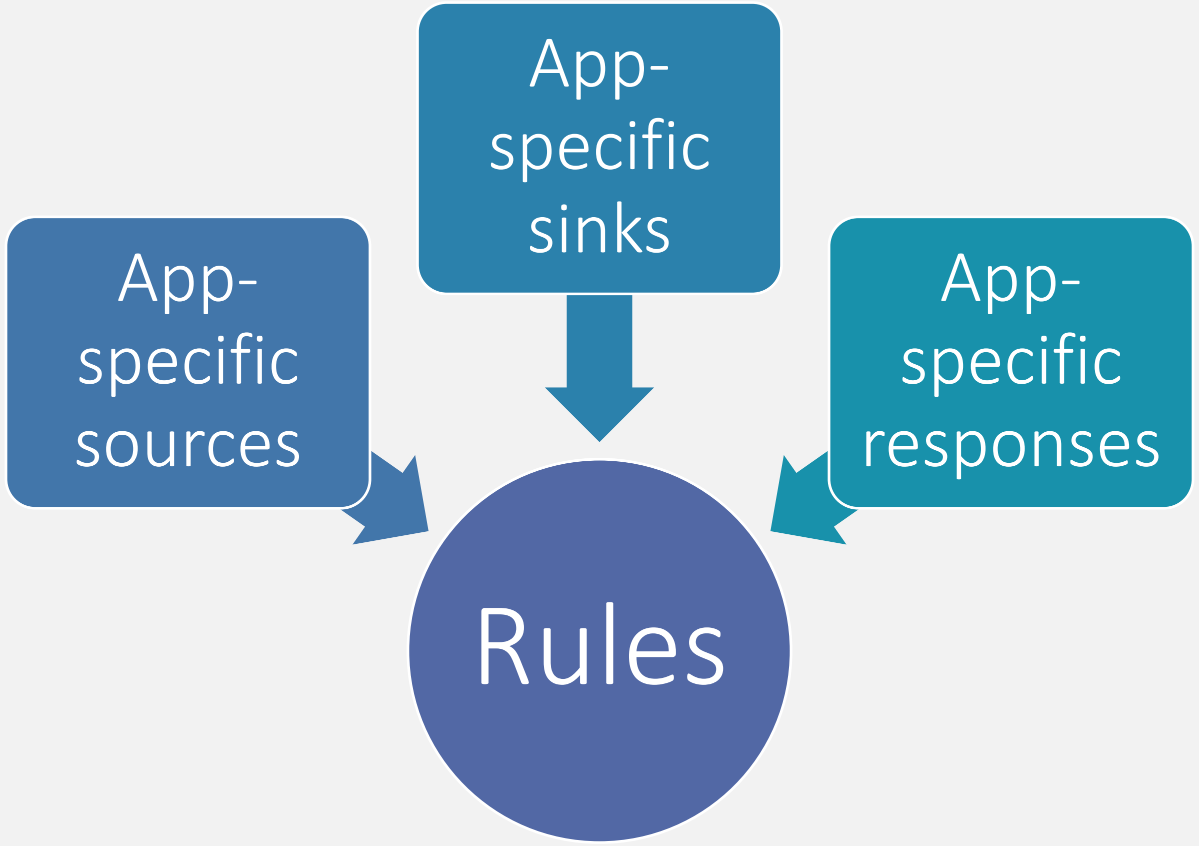
Automatically track tainted data

Manually verify sanitizers and transformations

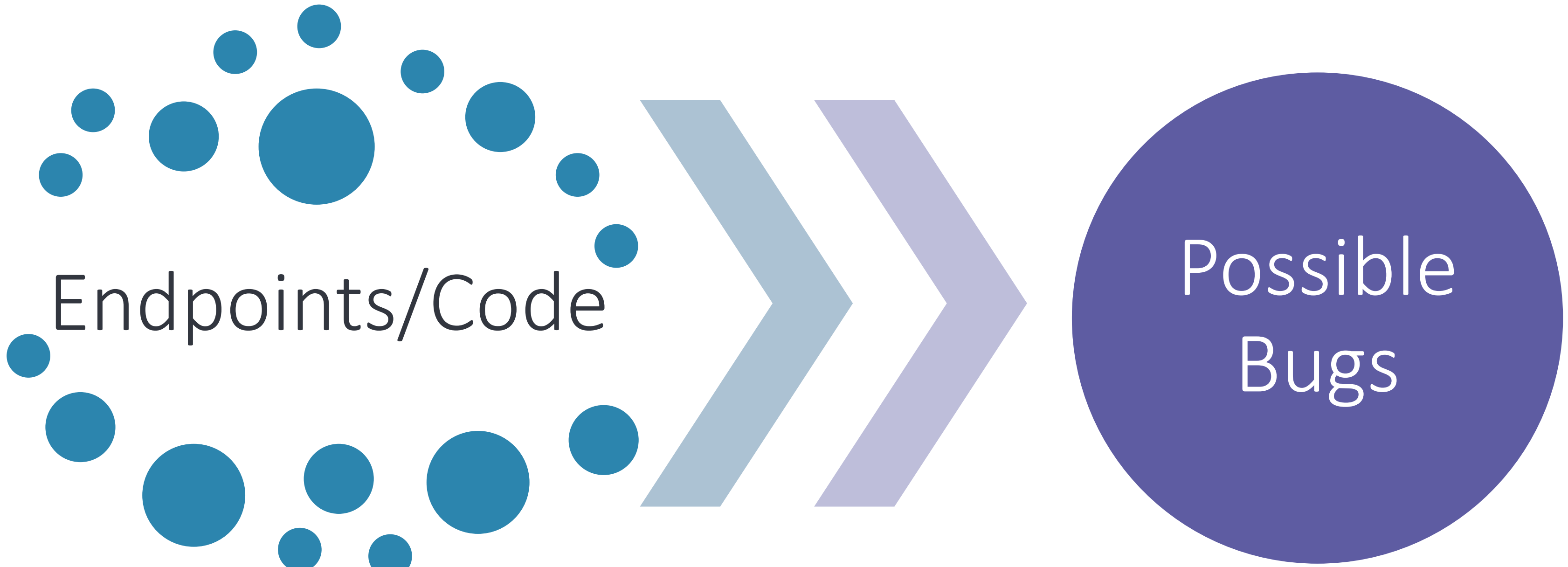
Manually write proof-of-concept

LOTS OF FALSE POSITIVES

The secret sauce is your curated custom rules. DON’T “spray and pray”!



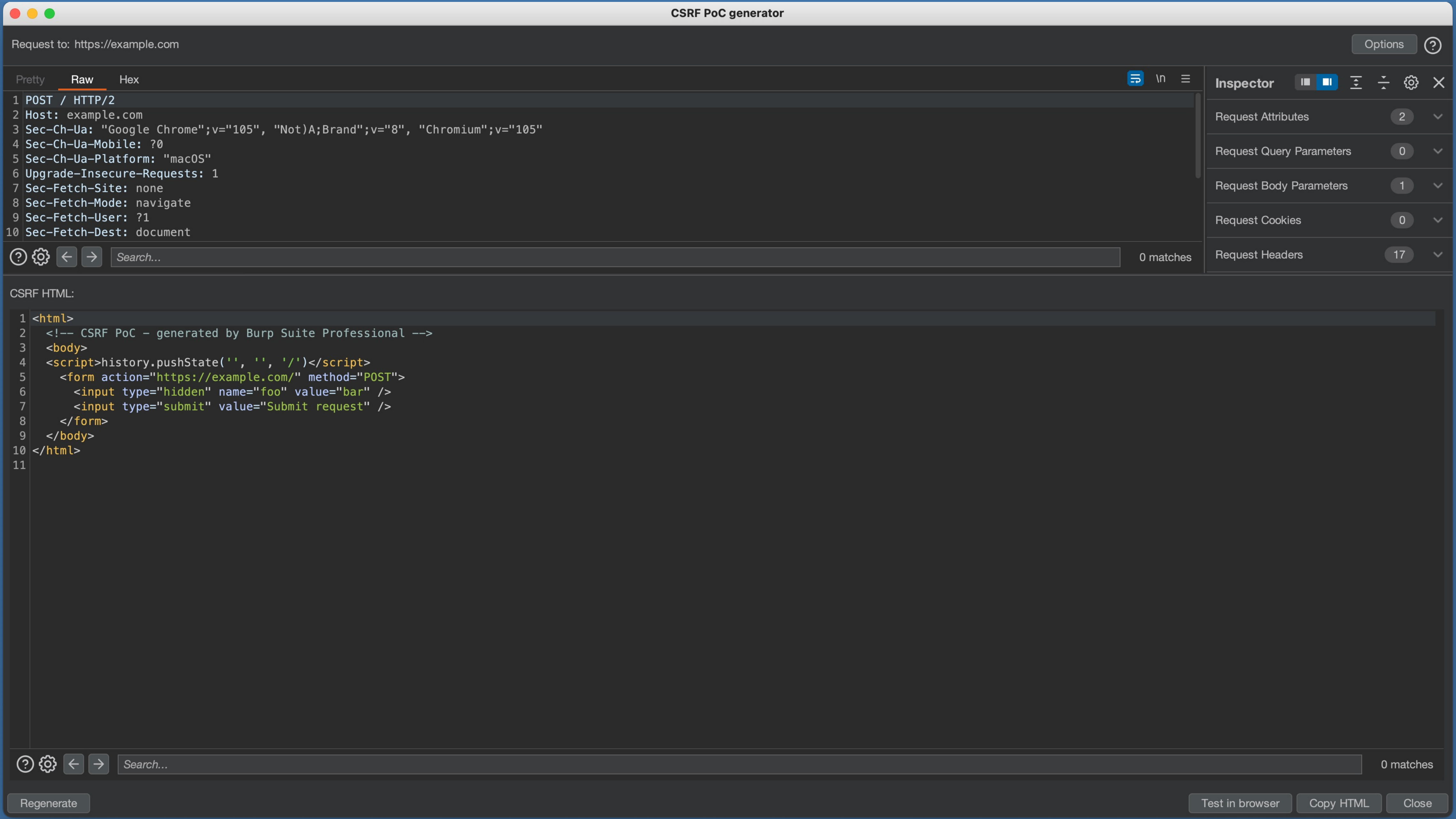
Accumulate your corpus of battle-tested rules over time.



Filtered automatically by SAST/DAST

Triaged manually by you

Exploit



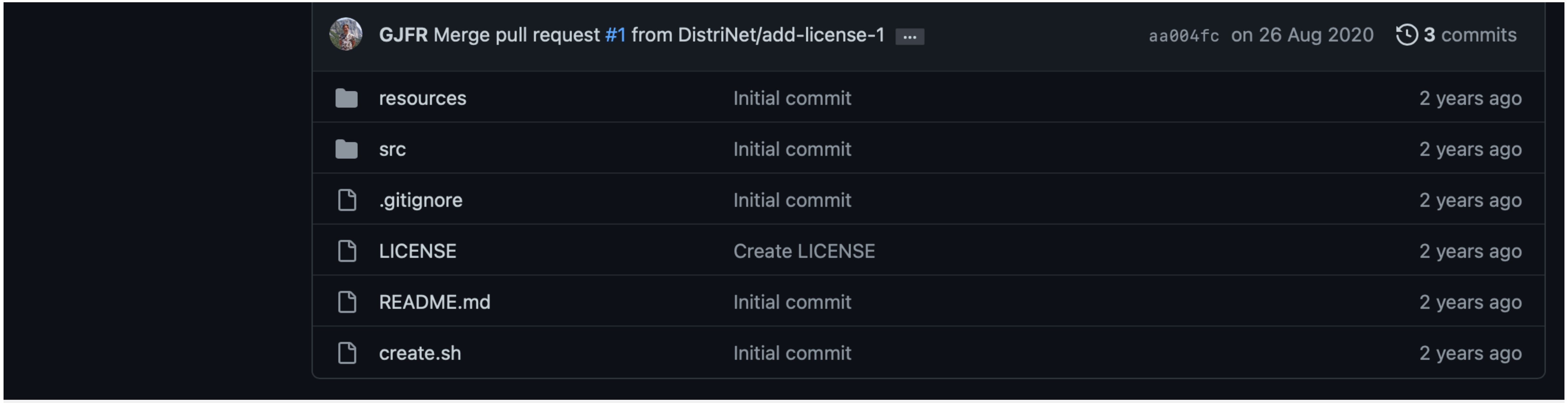
```
POST / HTTP/2
Host: example.com
Sec-Ch-Ua: "Google Chrome";v="105", "Not)A;Brand";v="8", "Chromium";v="105"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Upgrade-Insecure-Requests: 1
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
```

```html
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://example.com/" method="POST">
      <input type="hidden" name="foo" value="bar" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Case Study: Generating EPUB payloads

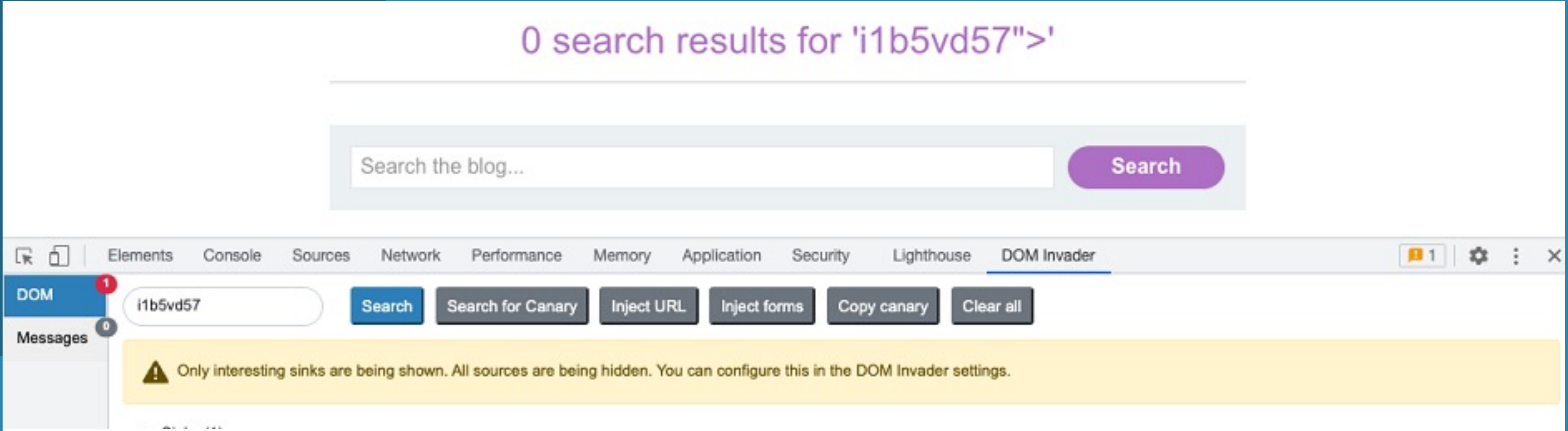
Don’t just copy and paste payloads, **generate** them.

- Locate exploit primitives e.g. HTTPLeaks
- Write generation scripts



GJFR Merge pull request #1 from DistriNet/add-license-1 ... aa004fc on 26 Aug 2020 3 commits

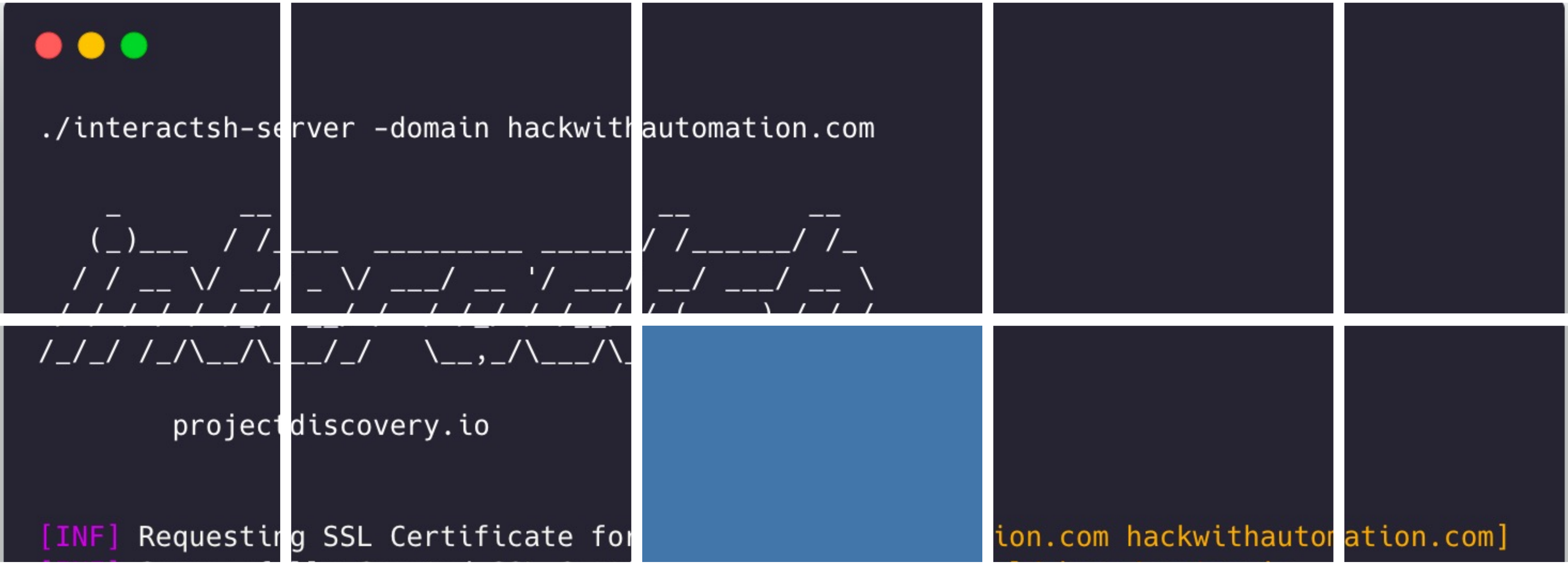
resources	Initial commit	2 years ago
src	Initial commit	2 years ago
.gitignore	Initial commit	2 years ago
LICENSE	Create LICENSE	2 years ago
README.md	Initial commit	2 years ago
create.sh	Initial commit	2 years ago



Only interesting sinks are being shown. All sources are being hidden. You can configure this in the DOM Invader settings.

Use tools like DOM Invader and write your own extensions.

Burp logs are one of your richest sources of data as a manual bug bounty hunter!

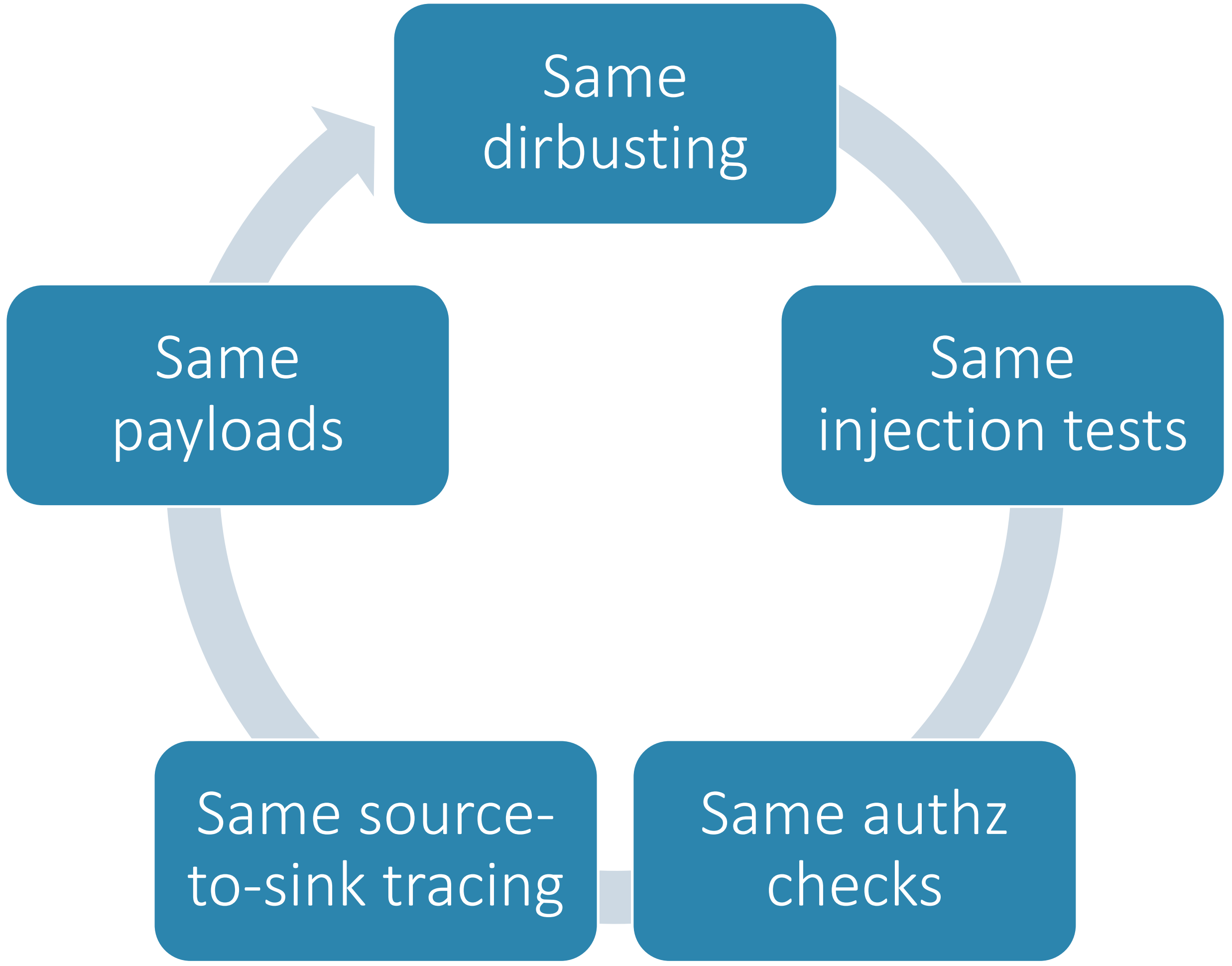


```
./interactsh-server -domain hackwithautomation.com

    _       __                         __       __
   (_)___  / /____  _________ ______/ /______/ /_
  / / __ \/ __/ _ \/ ___/ __ '/ ___/ __/ ___/ __ \
/_/_/ /_/\__/\___/_/   \__,_/\___/\_

        projectdiscovery.io

[INF] Requesting SSL Certificate for                    ion.com hackwithautomation.com]
```

You don’t need to spin your own recon framework but try creating and hosting your own exploit services.

Modern Manual Bug Hunting

If you find yourself doing the same thing again and again, it’s time to automate!



With the right automation, manual hunters can focus on their strengths. The goal is to complement your workflow, not replace it.

Targeted Recon

Focused Test

Comprehensive Exploit

Fun and Profit

Thank you

Blog: spaceraccoon.dev

Twitter: @spaceraccoonsec