

HUNTER



BOUNTY \$12,500



LVL 9

BUG BOUNTY HUNTING

HOW I GOT STARTED AND HOW YOU CAN TOO

breachsimrange

RECON

REPORT

REWARD



Security And Hacking Conference



FAHAD FAISAL @the_fahad_faisal

<https://breachsimrange.io>



PLAYER SELECT

MY BACKGROUND



SELF-TAUGHT

Started by reading disclosed HackerOne reports when studying on high school.

BUGS IN BIG NAMES

Found bugs in programs including Apple, Department of Defence, Dell, Spotify, Bitgo and my more.

STILL HUNTING

Still doing part-time bug bounty hunting. Earned over \$100k in rewards over different platforms.



ACCEPTED 31



BOUNTY \$12,500

PAID \$\$\$



TUTORIAL

WHAT IT IS

★ Companies pay you to find and responsibly report security bugs. The work is legal, remote, and paid per valid bug based on the severity

1

HUNT

Test an in-scope target

2

FIND

Discover a real bug

3

REPORT

Write it up with proof

4

REWARD

Valid bug earns a bounty



BUGS 47



BOUNTY \$12,500



CRIT



TUTORIAL

KEY TERMS



PLATFORMS

HackerOne, Bugcrowd and Intigriti list programs and pay bounties.

SCOPE

The assets you are allowed to test. Out of scope stays untouched.

SEVERITY

Rated low to critical. Higher impact means a higher bounty.

WORKFLOW

Submit, a triager validates, then a bounty to retest.



MINDSET

NARROW AND WIDE SCOPE



NARROW SCOPE

Specific assets like `app.example.com` and `api.example.com`.

Widely tested, so more competition.

WIDE SCOPE

Anything under `*.example.com`, plus third-party domains, mobile apps and GitHub leaks.

Find forgotten assets that get less attention.

LEVEL 1

MY ORIGIN STORY

It started with one disclosed report.

While going through the daily disclosed HackerOne reports, something caught my attention: cache-poisoning bugs by Youstin.



★ REP 9000

💰 BOUNTY \$12,500



STREAK x12

THE SPARK

DISCLOSED HACKERONE REPORTS



Exodus

High

Resolved

Cache Poisoning DoS on downloads.exodus.com

110

Bug reported by [youstin](#) was disclosed 5 years ago Uncontrolled Resource Consumption

A cache poisoning Denial of Service (DoS) vulnerability was discovered on downloads.exodus.com, which hosts files for download by Exodus users. The vulnerability was caused by a crafted Authorization header that resulted in a 403 error on the Azure storage host, which was then cached by Cloudflare and passed to all other users accessing the source. This could have resulted in a DoS attack by restricting users from downloading or accessing the files hosted on downloads.exodus.com. This summary was automatically generated.



HackerOne

Medium

Resolved

Static files on HackerOne.com can be made inaccessible through Cache Poisoning attack

228

Bug reported by [youstin](#) was disclosed 5 years ago Uncontrolled Resource Consumption

A vulnerability was discovered on hackerone.com that allowed an attacker to make static files inaccessible through a cache poisoning attack. The website used Cloudflare to cache static files, and an attacker could use the x-forwarded-scheme header to cause a redirect loop that would be cached by Cloudflare. By taking down a JavaScript file, an attacker could cause a total loss of availability on the website. The vulnerability affected any file cached on hackerone.com/*, including images, CSS files, and JavaScript files. This summary was automatically generated.



GitLab

High

Resolved

Cache poisoning Denial of Service affecting assets.gitlab-static.net

83

Bug reported by [youstin](#) was disclosed 5 years ago Uncontrolled Resource Consumption

A cache poisoning Denial of Service vulnerability was discovered in assets.gitlab-static.net, which is used to host JS and CSS files for gitlab.com. The vulnerability allowed an attacker to poison the cache and cause an empty response, leading to a denial of service on gitlab.com and all gitlab assets that use the specific CDN. The vulnerability affected all instances of gitlab where the CDN was used for JS and CSS files. This summary was automatically generated.



Rockstar Games

Medium

Resolved

Cache Poisoning DoS on updates.rockstargames.com

77

Bug reported by [youstin](#) and [d0xing](#) was disclosed 5 years ago Collaboration Violation of Secure Design Principles

A cache poisoning vulnerability was discovered on updates.rockstargames.com due to an unkeyed header, `trailer`. An attacker could exploit this weakness by sending a malformed response with status code 400, causing the cache to save it. The issue was resolved by working with the CDN provider to prevent the host from returning/saving a 400-code response when this header is sent in a request. This summary was automatically generated.

youst.in/research/cache-poisoning-at-scale

<https://breachsimrange.io>



BUG BOUNTY

WHAT A WEB CACHE IS

*A cache stores a copy of a page the first time it is served, so the next visitor gets that copy instead of reaching the origin server.

USER A

Requests the page. Origin builds it, cache stores a copy.

CACHE

Holds the copy close to users.

USER B

Gets the cached copy instantly.

* If an attacker makes the cache store a harmful or broken response, every later visitor is served it from the cache.

★ THE FIRST WIN

★ MY FIRST BUG

★ 1 MANUAL FIRST

Tested my private programs one by one, by hand.

★ 2 SIMPLE SCRIPT

It checked a single condition, nothing more.

★ 3 IT WORKED

Enough to land my first valid report.

```
test.py
10
11
12 domain = input(Fore.GREEN + "The domain need to test: ")
13
14 domain_scan = "https://" + domain + "/favicon.ico?version=poc123"
15
16 default_headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0', 'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'}
17
18 cookies = {'Cookie': 'amp_ddd6ec=6LwNeSYMZM-n7G2uV7Arh8...1frqdddb9.1frqdo4fu.12.15.27; _gcl_au=1.1.1342798764.1641930296; _ga=GA133182921.15462798764.1641930296.1641930296.1641930296'}
19 session = requests.Session()
20 session.max_redirects = 10
21 sending_request = session.get(url = domain_scan, headers = default_headers, cookies = cookies , allow_redirects=False)
22 space_line()
23 print(Fore.BLUE + "The status code : " + str(sending_request.status_code))
24 space_line()
25 print(Fore.RED + "The url used for testing : " + sending_request.url)
26 space_line()
27 print("Response Headers : " + str(sending_request.headers))
28 response_headers = sending_request.headers
29 status_code = sending_request.status_code
30
31 try:
32     sending_request = session.get(url = domain_scan, headers = default_headers, cookies = cookies , allow_redirects=True)
33
34 except requests.exceptions.TooManyRedirects:
35     space_line()
36     print(Fore.GREEN + "Target Is vulnerable")
37
```

I created a python script to scan all the targets for this issue and found this on many private programs.

TARGETS 128



BOUNTY \$12,500



LIVE 94

BUG BOUNTY

x-forwarded-sche → 301 → 301 → 301 → ...

```
test.py
10
11
12 domain = input(Fore.GREEN + "The domain need to test: ")
13
14 domain_scan = "https://" + domain + "/favicon.ico?version=poc123
15
16 default_headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0' , 'Accept': 'text/html,ap
17
18 cookies = {'Cookie': 'amp_ddd6ec=6LwNeSYMZM-n7G2uV7Arh8...1frqdddb9.1frqdo4fu.12.15.27; _gcl_au=1.1.1342798764.1641930296; _ga=G
19 session = requests.Session()
20 session.max_redirects = 10
21 sending_request = session.get(url = domain_scan, headers = default_headers, cookies = cookies , allow_redirects=False)
22 space_line()
23 print(Fore.BLUE + "The status code : " + str(sending_request.status_code))
24 space_line()
25 print(Fore.RED + "The url used for testing : " + sending_request.url)
26 space_line()
27 print("Response Headers : " + str(sending_request.headers))
28 response_headers = sending_request.headers
29 status_code = sending_request.status_code
30
31 try:
32 | sending_request = session.get(url = domain_scan, headers = default_headers, cookies = cookies , allow_redirects=True)
33
34 except requests.exceptions.TooManyRedirects:
35 | space_line()
36 | print(Fore.GREEN + "Target Is vulnerable")
37
```

SCALING UP

AUTOMATING THE CHECK

```
def domain_scanner(sub_domnames):
    print(Fore.RED + '-----Scanner Started-----')
    space_line()
    print(Fore.BLUE + '-----URL after scanning domains-----')

    # loop for getting URL's
    for subdomain in sub_domnames:
        # making url by putting subdomain one by one
        url = f"https://{subdomain}/robots.txt?version=poc123"
        # using try catch block to avoid crash of
        # the program
        try:
            # sending get request to the url
            request_send = requests.get(url , headers = default_headers, cookies = cookies, timeout=6, allow_redirects= False)

            # if url is invalid then pass it

        except requests.exceptions.ReadTimeout:
            pass

        except requests.ConnectionError:
            pass

        except requests.exceptions.InvalidURL:
            print(Fore.BLUE + 'An invalid url found : ' + url)

        except KeyboardInterrupt:
            print(Fore.RED + "Exiting")
```

THE BUG EXISTS ON MANY TARGETS

LOOT

BOUNTIES FROM EVERYWHERE

- 

Redacted rewarded [cametome006](#) with a \$350 bounty.

November 7, 2022, 11:49pm UTC

Hi @cametome006,

Please accept our apologies for the delayed response. We will do better going forward, and we are awarding this bounty with a bonus to recognize your patience and understanding.
- 

Redacted rewarded [cametome006](#) with a \$400 bounty and a \$100 bonus.

July 7, 2022, 2:08pm UTC

@cametome006 thanks for this report -- it's a very creative one, and we really appreciate your work here. We will get it fixed as soon as possible!
- 

Redacted rewarded [cametome006](#) with a \$500 bounty.

May 6, 2022, 7:47pm UTC

Hey @cametome006 Thanks for your efforts finding this vuln! We appreciate it!

⋮
- 

Redacted rewarded [cametome006](#) with a \$1,000 bounty.

Thank you for your report. We validated the vulnerability and have rolled-out a fix.
- 

Redacted rewarded [cametome006](#) with a \$300 bounty.

April 28, 2022, 9:25am UTC

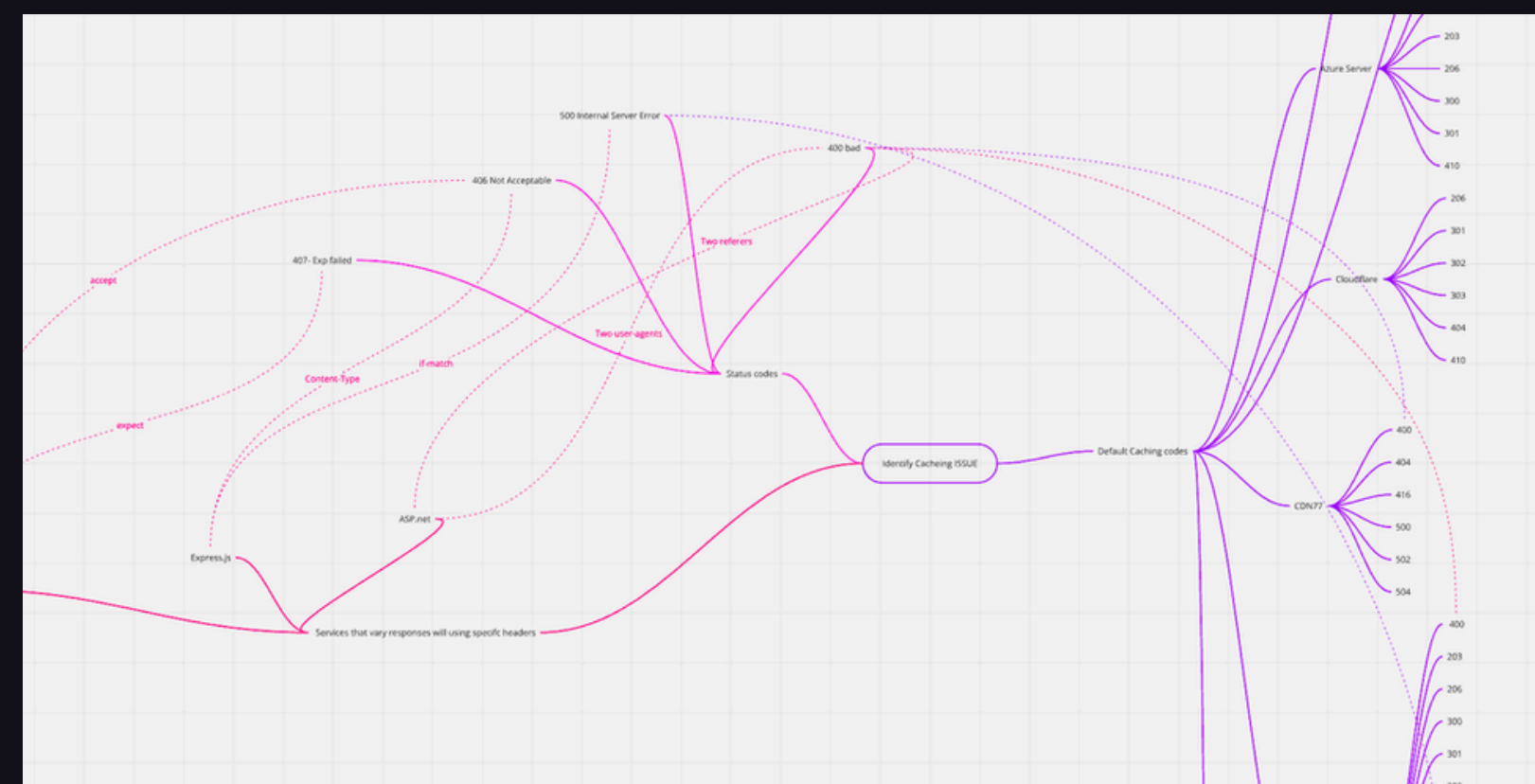


GOING DEEPER

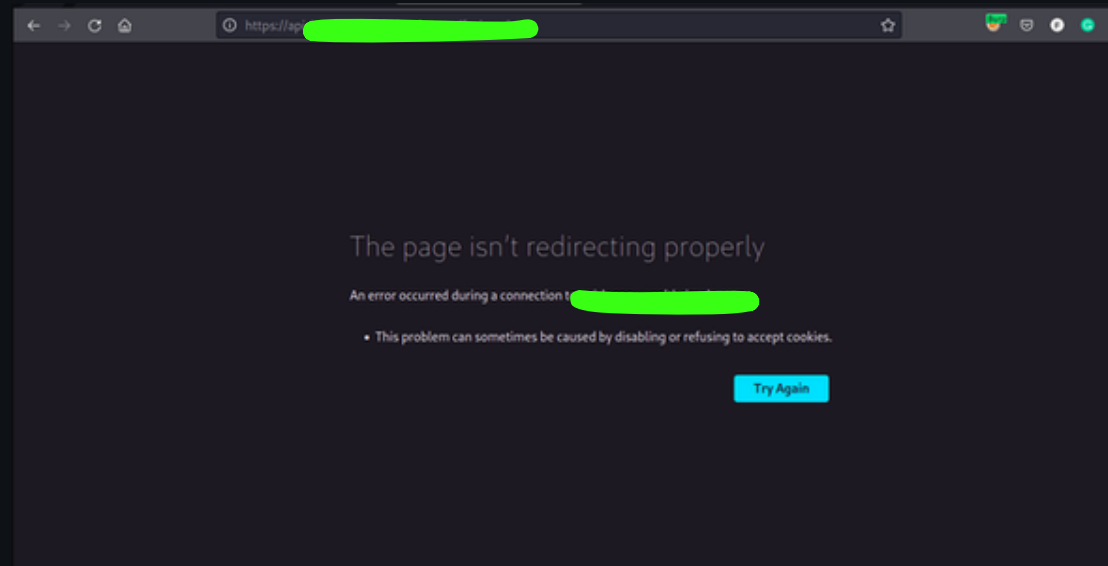
ONE BECAME MANY

★ VARYING ONE PATTERN

- + Different CDNs: Akamai, CloudFront, Cloudflare
- + Malformed or duplicate headers
- + Responses cached by mistake
- + Automation of each new pattern



EVIDENCE



Redirect-loop DoS in browser

cacheing the 301 redirect loop
leading to static DOS

```
GET / HTTP/2                                HTTP/2 400 Bad Request
Host: www.redacted.com                      X-Check-Cacheable: YES
: 1
```

Cacheing 400 bad request response

cacheing the 400 bad request
response if illegal value like / is
sent in the request headers

LOOT // RESOLVED AND PAID

REWARDS from many programs

Cache poisoning denial of service by adding an invalid character per RFC 7230 across various programs, resolved and paid.

CPDOS due to misconfig

 In progress - Submitted 31 Jan 2023 - Last activity a year ago

P3 Resolved

\$750

10 points

Comment 1

Cache Poison DOS on  adding invalid character defined in RFC7230 (/) in the request.

 In progress - Submitted 28 Oct 2022 - Last activity 2 years ago

P3 Resolved

\$750

10 points

Comments 7

Cache Poison DOS on  domains by adding invalid character defined in RFC7230 (/) in the request.

 In progress - Submitted 28 Oct 2022 - Last activity 2 years ago

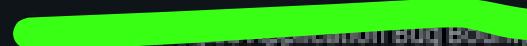
P3 Resolved

\$0

10 points

Comments 8

Cache Poison DOS on  by adding invalid character defined in RFC7230 (/) in the request.

 In progress - Submitted 22 Sep 2022 - Last activity 2 years ago

P3 Resolved

\$750

10 points

Comments 4

ACCEPTED 31

BOUNTY \$12,500

PAID \$\$\$

LEVEL 2

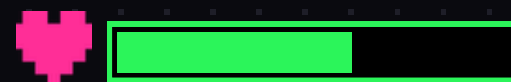
HOW I HUNT

My repeatable recon process.



Manual → AI ASSISTED

<https://breachsimrange.io>



THE PIPELINE

RECON PIPELINE

1

ENUMERATE

Pull subdomains

haktrails anew

2

PROBE

Find live hosts

httpx

3

COLLECT

Historical URLs

gau wayback

4

MATCH

Flag likely bugs

gf

Then open the interesting URLs and review them by hand. Tools surface candidates, manual review finds the bug.

THE REAL COMMANDS

RECON IN THE TERMINAL

```
~ A Pair Ctrl I Dispatch Beta Ctrl Shift I  
cat scope | haktrails subdomains | anew subs.txt
```

haktrails subdomains

```
~ A Pair Ctrl I Dispatch Beta Ctrl Shift I  
cat subs.txt | ~/go/bin/httpx -sc -cl -ct -title -server -td | anew hosts/http.out
```

httpx live hosts

```
~ A Pair Ctrl I Dispatch Beta Ctrl Shift I  
cat subs.txt | cut -d " " -f1 | gau --threads 5 | tee urls.txt
```

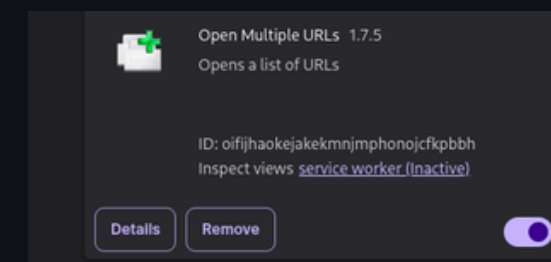
gau historical URLs

```
~ A Pair Ctrl I Dispatch Beta Ctrl Shift I  
gf xss urls.txt | tee xss_candidates.txt  
gf ssrf urls.txt | tee ssrf_candidates.txt  
-a →
```

gf pattern match

```
~ A Pair Ctrl I Dispatch Beta Ctrl Shift I  
cat xss_candidates.txt | pbcopy 1.txt →
```

pbcopy one-liner



open interesting URLs

GO DEEPER

AUTHENTICATED HUNTING

★ Once you have targets, choose an app with a login. Private dashboards, billing and team management hold the higher-impact bugs.

1 LOG IN

Use test credentials on an authenticated area

2 PROXY ALL

Route all traffic through Burp Suite

3 WATCH THE API

See how the frontend talks to the backend, then tamper

★ Extensions I keep on: Burp Flow for readable history, AutoRepeater for firing tampered requests and most **IMPORTANTLY!!!!** : Read the docs

GO DEEPER

JAVASCRIPT ANALYSIS

★ The frontend JavaScript reveals internal APIs, hidden parameters and undocumented endpoints, all from the client side. ★

1

COLLECT THE JS

Grab every .js file through Burp while browsing, plus waybackurls for older ones.

2

EXTRACT ENDPOINTS

Run xnLinkFinder over the JS to pull URLs, parameters and API paths.

3

READ IT

Beautify, open in VS Code, search for userId, token and fetch() calls.

REAL WORLD

DEALING WITH TRIAGERS: TIPS

Hi msfvenom,

Thank you for your reply. After further reviewing the report we believe that the rate limiting kicks in after a while making the 400 Bad Request to poison only for short duration.

Hence this would not have any impact here. Could you please elaborate on this and if you are able to bypass it somehow ?

They asked for real impact, and whether I could bypass the rate limit.

☒ Create new resource pool

Name: Custom resource pool 1

☐ Maximum concurrent requests:

☒ Delay between requests:

10000

milliseconds

☒ Fixed

☐ With random variations

A reproducible proof of concept video that showed measurable impact.

- + Triagers may get confused and that is normal
- + They want you to demonstrate impact
- + A reproducible, measurable result settles it
- + Proof of concept script is better

LOOT // RESOLVED AND PAID

ISSUE I FOCUS: IDORS and BACs

IDORs and RBACs in the APIs reached other tenants' confidential documents, 2fa bypasses and ATO

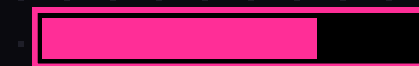
[REDACTED] Data [REDACTED] In progress • Submitted 15 Apr 2025 • Last activity 8 days ago P3 Unresolved	\$750 10 points Comment 1
[REDACTED] [REDACTED] In progress • Submitted 04 Apr 2025 • Last activity 14 days ago P3 Unresolved	\$750 10 points Comments 8
[REDACTED] [REDACTED] In progress • Submitted 25 Mar 2025 • Last activity 25 days ago P3 Unresolved	\$700 10 points Comments 2
[REDACTED] [REDACTED] In progress • Submitted 18 Mar 2025 • Last activity a month ago P2 Unresolved	\$2,500 20 points Comments 4

MORE!

HUNTER



BOUNTY \$12,500



LVL 9

LEVEL 3

MY AI AGENT SETUP

A central **API** that keeps all the context in one place, multiple agents running in parallel for different tasks and a **tunnel**, so I can access everything from anywhere.





AI SETUP

WHY AI IS HERE

★ Recon and first-pass triage are repetitive. Agents handle that volume, so I spend my time on judgment. ★

AI HANDLES

Enumerate, crawl, diff, summarise JS,
flag candidates.

I HANDLE

Pick targets, verify impact, craft the
PoC and report.

AI SETUP

MY STACK >



One hunting agent, a skills library, and a shared notes API.

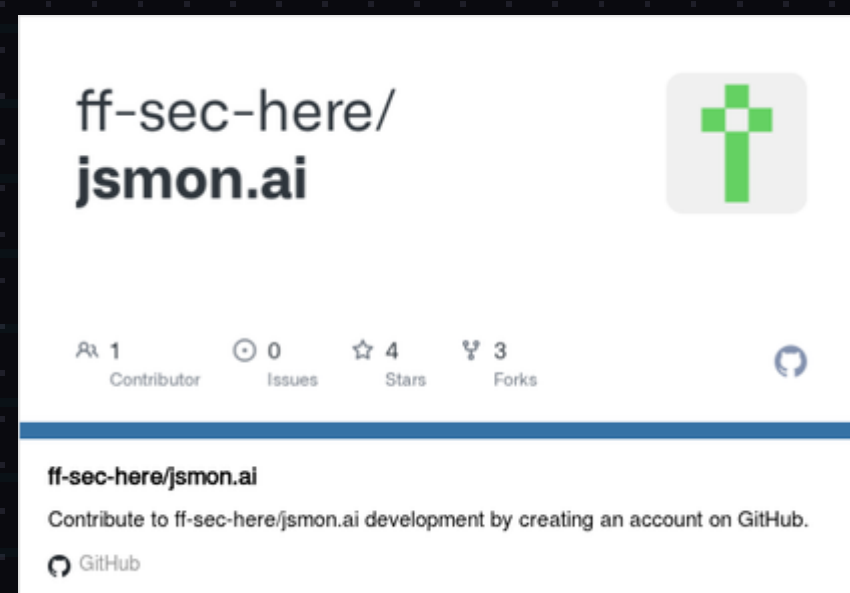
- > Driver: Claude Code (also used kiro-cli)
- > Skills: recon, fuzzing, portscan, js-recon, burp
- > BB-Mapper API keeps every target and subdomain note
- > A second agent dumps each session back into the notes

AI SETUP

JS MONITORING

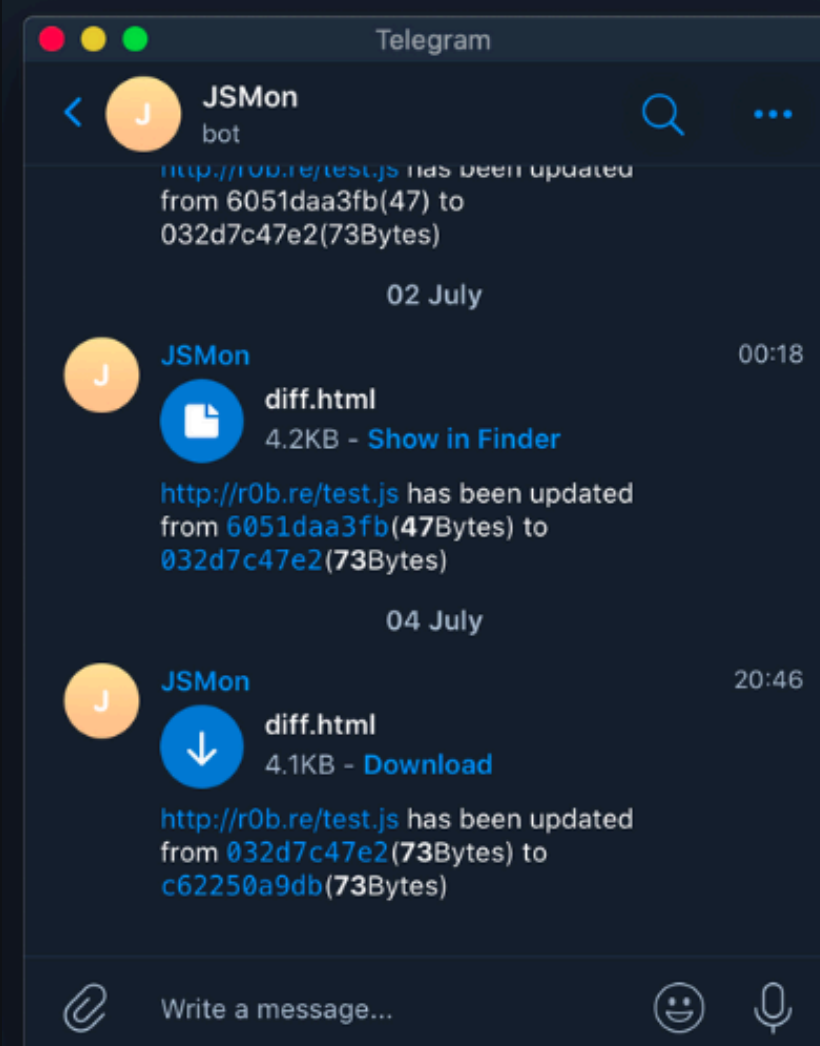
It watches target JavaScript and tells me when a
⭐ change looks risky.

- > Fetches and hashes each JS file
- > Gemini scores every change
- > HIGH risk pings Telegram

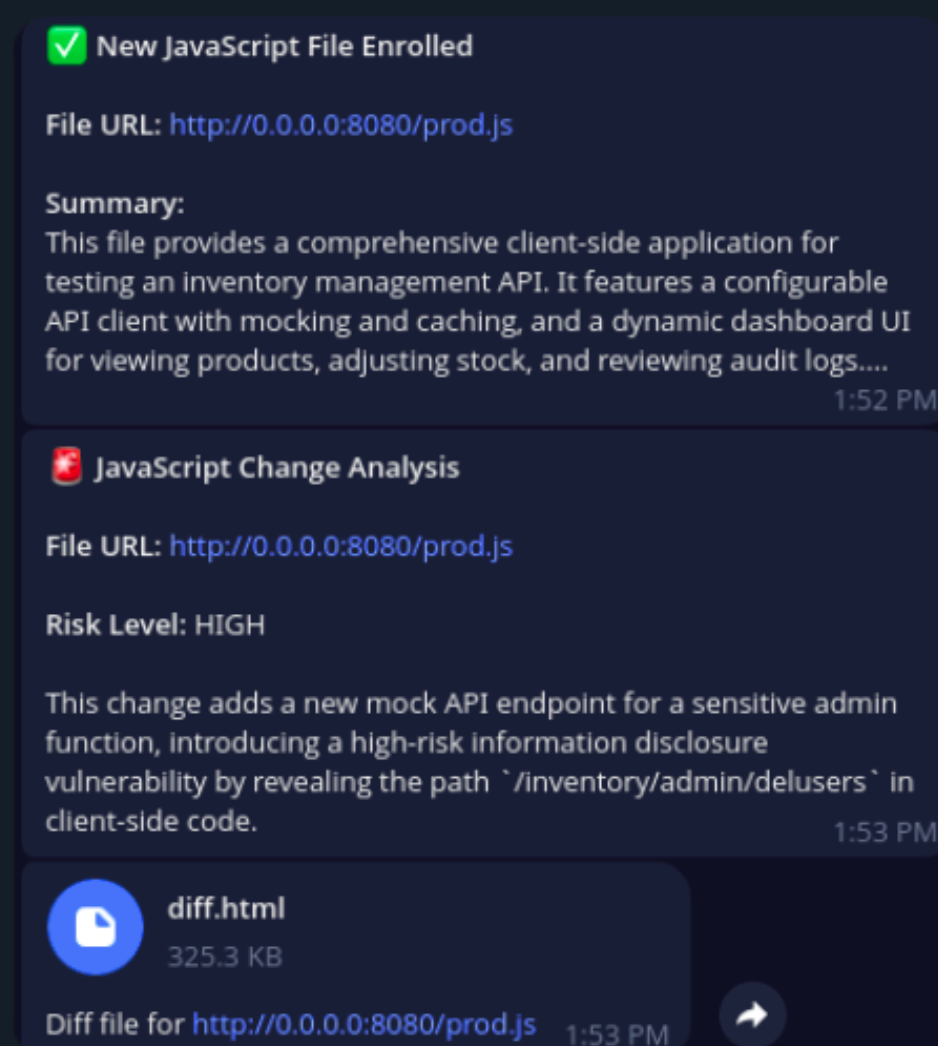


JSMon.ai watches target JavaScript and alerts me when a change looks high risk, with a diff and a plain summary.

JSMON



JSMON.AI



AI SETUP

THE MAPPER

Every agent reports into one place. The mapper holds notes, subdomains, JS files, reports and checklists for each target.



Targets

Alerts

Checklists

Progress

Scanning

Settings

←

test ●

Checklists

Shell

Export JSON

Delete

Overview ▾

Assets ▾

Recon ▾

Knowledge ▴

Sessions

Notable

Notes

Files

Scope Entries

Add Scope Entry



AI SETUP

TARGET NOTES

Years of my own notes become the knowledge base: findings, leads and context the agents read and write back.

Overview

Assets

Recon

Knowledge

Sessions

Target Notes

General notes for this target

Edit Notes

Findings & Leads

Subdomain	Severity	Title	Status
			Lead
			Suspicious

Global Context

AI SETUP

IT WATCHES FOR CHANGES

When a target changes, an agent checks if it matters. A status code moving from 200 to 403 once exposed an internal data leak I would have missed.



Alerts Dashboard

Monitor changes in your reconnaissance data across all targets

Clear Visible (1)

Clear All

Refresh

• TOTAL ALERTS

19

All-time monitoring

• UNREAD

13

Requires attention

• NOTABLE

13

High priority

Showing 1 - 1 of 1 alerts

STATUS CHANGED NEW

Status changed: legacy.testcorp.com went from 200 to 403

test

20h ago

• PREVIOUS STATE

status_code: 200

• NEW STATE

status_code: 403

Acknowledge

Mark Notable

View Resource

Add Notes

AI SETUP

THE SKILLS LIBRARY

Each skill is a reusable play the agent can run: recon, fuzzing, portscan, js-recon, burp, 401-403 bypass, bug-bounty-mapper.

401-403-bypass-skill	Add Apache HTTP Server Confusion Attacks section (Orange Tsai 2024)
bug-bounty-mapper-skill	change to bbm cli
burp-project-parser-skill	Add burp project file parser skill
burp-skill	Initial commit: agent configs, skills, prompts, and scripts
fuzzing-skill	Remove Burp mentions from all skills/prompts except burp-skill
js-recon-skill	Initial commit: agent configs, skills, prompts, and scripts
portscan-skill	Initial commit: agent configs, skills, prompts, and scripts
recon-skill	Initial commit: agent configs, skills, prompts, and scripts

AI SETUP

A SKILL FILE

Each skill is a small SKILL.md with a strict scope. This one, bug-bounty-mapper, lets the agent talk to the mapper over its bbm CLI.

```
head bug-bounty-mapper-skill/SKILL.md
```

```
---
```

```
name: bug-bounty-mapper
```

```
description: >
```

```
  Use the `bbm` CLI to interact with BB-Manager at [REDACTED]  
  Use when the user wants to manage bug bounty data: targets, subdomains, notes,  
  checklists, scans, fuzzing, ports, alerts, dorking, JS recon, and more.
```

```
---
```

```
# Bug Bounty Mapper – bbm CLI Skill
```

AI SETUP

THE BRAIN-DUMP AGENT

A second agent reads the session log, sorts what was done by category, and writes structured notes back per target and subdomain.

```
structured target main note that helps future hunting sessions pick up with full context.

You use the bug-bounty-mapper-skill for all BB-Mapper API interactions.

## Your Inputs

- `BB_SESSION_ID` - kiro conversation id of the active bugbounty session
- `BB_TARGET_ID` - BB-Mapper target ID (detect from session if missing)
- `BB_LAST_DUMP_TS` - last run timestamp, stored in `/tmp/bb-braindump-last.ts`

## Step 1: Load Session Log

```bash
sqlite3 ~/.local/share/kiro-cli/data.sqlite3 \
 "SELECT value FROM conversations_v2 WHERE conversation_id='$BB_SESSION_ID' ORDER BY updated_at DESC LIMIT 1;"
```

Parse `history`. Filter to entries newer than `BB_LAST_DUMP_TS`. Each entry has `user.content.Prompt.prompt` and `assistant` fields.

## Step 2: Extract What Was Done

Identify actions by category:

Category	Signals
**recon**	subfinder, amass, httpx, subdomain enumeration
**fuzzing**	ffuf, wordlist, directory/file/vhost fuzzing
**portscan**	nmap, masscan
**js-recon**	JS file analysis, secrets, endpoint discovery
**manual-testing**	HTTP requests, parameter testing, auth testing
**finding/lead**	vulnerability confirmed or suspected

For each action extract: what was done, which subdomain(s), any results or findings.

## Step 3: Fetch Existing Notes (dedup check)

Before writing anything, fetch current notes:
- `GET /api/targets/{id}/notes/main` - target main note
- `GET /api/subdomains/{id}/note` - for each relevant subdomain

Skip anything already documented.

## Step 4: Write Subdomain Notes

For anything specific to a subdomain (recon results, fuzzing hits, port findings, JS secrets, leads), append to that subdomain's note:
```

THE PAYOFF

BUG TYPES I FIND

DURING RECON

- + Leaked API keys in GitHub repos
- + Open S3 buckets with sensitive files
- + Misconfigured exposed services

INSIDE THE APP

- + IDOR, accessing other users' data
- + Broken access control and priv-esc
- + Business-logic flaws in payments

THE PAYOFF

- > Auth bypasses, IDORs, logic bugs (mostly) cache poisonings, RBACs and many more

<https://breachsimrange.io>



AI SETUP

HUMAN IN THE LOOP

★ AI JUST AMPLIFIES YOUR SKILLS.

VERIFY EVERY LEAD

Reproduce it yourself before it counts as a bug.

IMPACT STAYS WITH YOU

You present the steps to the triager via video POC or images.

RESPECT THE PROGRAM

Avoid mass automated noise and always prefer quality.

TAKEAWAYS

LESSONS LEARNED

1

SPECIALISE

Choose one issue that interests you and go deep.

2

KEEP NOTES

Track targets and techniques so you can reuse them.

3

CLEAR REPORTS

A clear proof of concept often earns extra rewards.

4

ASK THE COMMUNITY

When a bypass has you stuck, others often help.



BUGS 47



BOUNTY \$12,500



CRIT



YOUR TURN

YOUR FIRST WEEK



1

PICK A PLATFORM

Make a HackerOne or Bugcrowd account today

2

PICK ONE BUG CLASS

take a bug class and study that in depth

3

READ REPORTS DAILY

Disclosed reports is your free curriculum

4

SET UP YOUR KIT

Build up your arsenal using your own workflow

5

SUBMIT A REPORT

Your first one, even a duplicate teaches the loop





THANK YOU

QUESTIONS?

PING ME

@came tome006

@the_fahad_faisal

[linkedin.com/fahadsec](https://www.linkedin.com/fahadsec)