# Unknown Title

Hacking APIs: Workshop

#whoami

Corey Ball

@hAPI_hacker

- 12+ years in IT & Cyber
- Cyber Security Consulting Senior Manager, Moss Adams
- Author of Hacking APIs (No Starch Press, 2022)
- Apisec.ai Evangelist
- Creator of the APIsec University (apisecu.com)

Preparing Your API Hacking System

For the demonstration of this workshop I will be using Kali Linux with:

- Postman
- Burp Suite
- FoxyProxy
- MITMProxy2Swagger

https://bit.ly/3QEu9me

https://github.com/hAPI-hacker/Hacking- APIs

Goals

1. Find APIs
2. Reverse Engineer Docs
3. Exploit:
    1. Excessive Data Exposure
    2. Broken Object Level Authorization

What are APIs?

Application Programming Interfaces, but that doesn't help.

APIs are a technology that facilitates a common method for applications to communicate.

Metaphor:

- Legos

Different Types of REST APIs:

- Public: Easiest to find
- Partner: More challenging
- Private: Most challenging

API Vulnerabilities for Breakers

1. Authorization (BOLA +BFLA)
2. Authentication
3. Excessive Data Exposure
4. Improper Assets Management
5. Mass Assignment
6. Security Misconfiguration
7. Insecure Design / Business Logic Flaws
8. Lack of Rate Limiting

Discover APIs

In order to hack APIs, you must first be able to find them. Next I will go over how to uncover the API attack surface of a target using passive and active reconnaissance technique

Discovery GOALS:

- Find a live API
- Find API docs
- Find leaked info (Keys, tokens, etc.)

Finding Public APIs

Public APIs are advertised and marketed to be found.

- Check the footer of a landing page for links like:

API, Developers, Dev, Resources, Docs, etc

- Use Non-1337 Google Hacking

Perhaps the API is not advertised and easy to find... then you can deploy passive reconnaissance.

Web API Indicators

But first you will need to know what you are looking for. There are several indicators that you should be aware of that may indicate that you've found an API.

- Obvious Naming Schemes
- Headers
- Responses
- 3rd Party Sources

Obvious Naming Schemes

https://target-name.com/api/v1

https://api.target-name.com/v1

https://target-name.com/docs

https://dev.target-name.com/rest

Look for API indicators within directory names like:

/api, /api/v1, /v1, /v2, /v3, /rest, /swagger, /swagger.json, /doc, /docs, / graphql, /graphiql, /altair, /playground

Also, subdomains can also be indicators of web APIs:

api.target-name.com

uat.target-name.com

dev.target-name.com

developer.target-name.com

test.target-name.com

HTTP Headers

"Content-Type: application/xml"

"Content-Type: application/json, application/xml"

Responses

{}

{"message": "Missing Authorization token"}

{"error": {"code": "VALIDATION_ERROR", "description": "Authorization is a required parameter.", "field": "Authorization", "instance": null}}

{"GibberishinJSON"}

Passive Recon

Obtaining information about a target's APIs without directly interacting with them. We are looking for evidence of APIs and leaked information.

- 3rd Party Sources
- Google + Git Dorking
- Wayback Machine
- Shodan

3rd Party Sources

- Github
- Postman Explore
- ProgrammableWeb
- RapidAPI

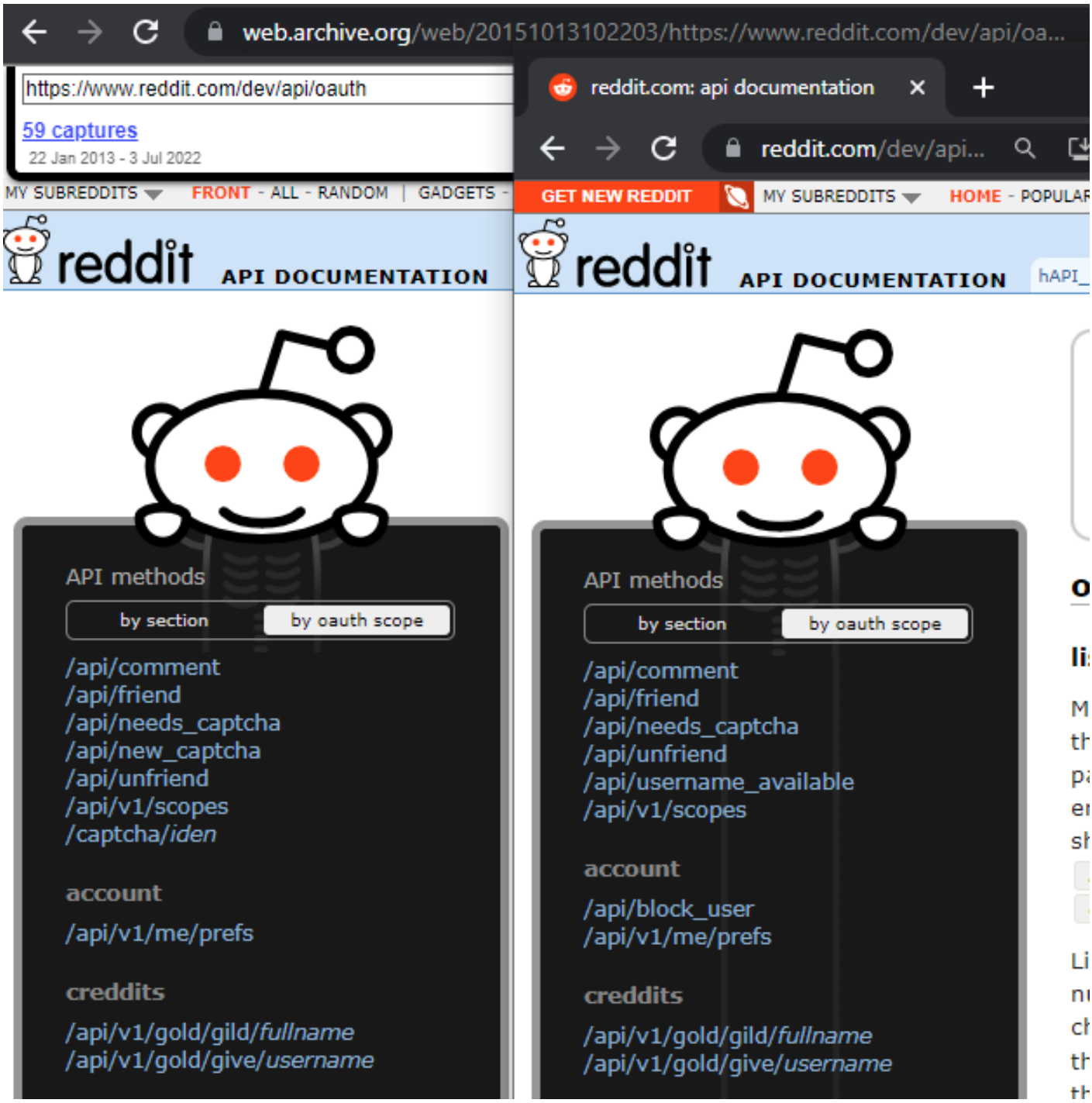
GitDorking

GitHub for your target organization’s name paired with potentially sensitive types of information, such as “api key,” "api keys", "apikey", "authorization_bearer", "access_token", "secret", or “token.”

Investigate the various GitHub repository tabs to discover API endpoints and potential weaknesses. Analyze the source code in the Code tab, find bugs in the Issues tab, and review proposed changes in the Pull requests tab.

Way Back Machine

- Compare current versions of API documentation with past versions




Shodan Queries:

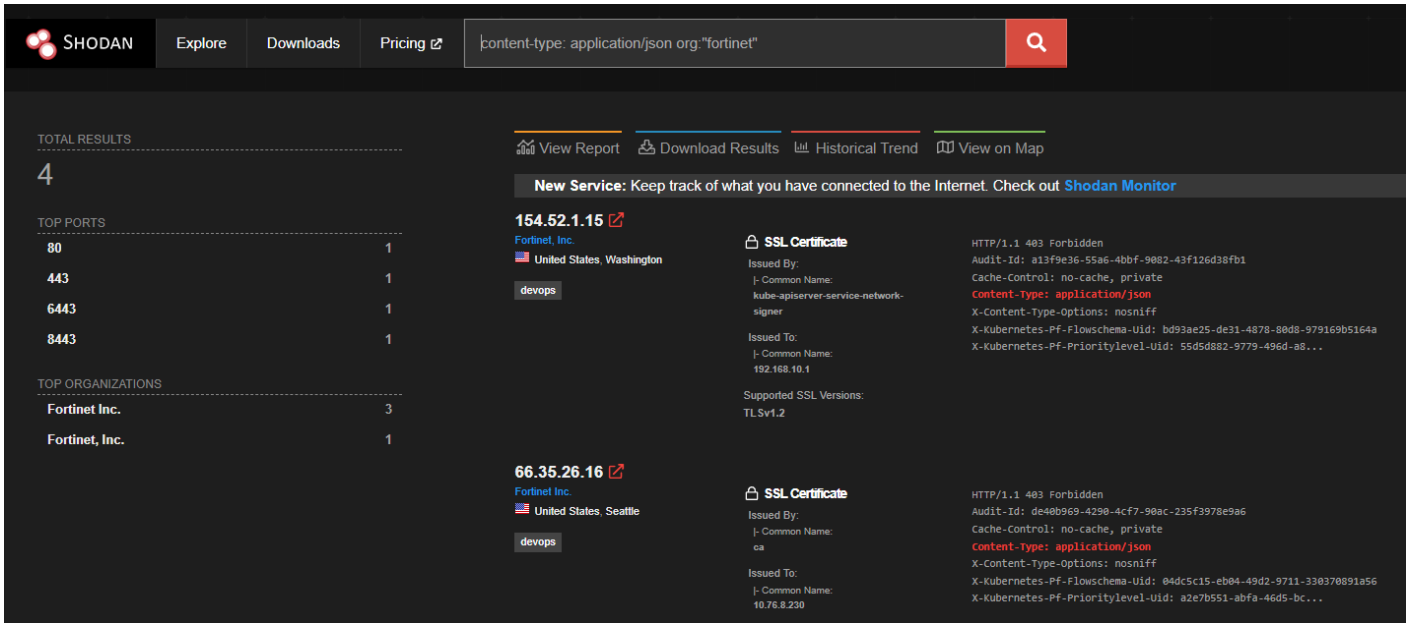
hostname:"targetname.com" : Using hostname will perform a basic Shodan search for your target's domain name. This should be combined with the following queries to get results specific to your target.

"content-type: application/json" : APIs should have their content-type set to JSON or XML. This query will filter results that respond with JSON.

"content-type: application/xml": This query will filter results that respond with XML.

"200 OK": You can add "200 OK" to your search queries to get results that have had successful requests. However, if an API does not accept the format of Shodan's request, it will likely issue a 300 or 400 response.

"wp-json": This will search for web applications using the WordPress API.




Active Recon

Active API discovery is the process of interacting directly with the target primarily through scanning the environment.

- Nmap
- OWASP Amass
- Gobuster
- Kiterunner

Nmap

- $ nmap -sC -sV [target address or network range] -oA nameofoutput
- $ nmap -p- [target address] -oA allportscan
- $ nmap -sV --script=http-enum <target> -p 80,443,8000,8080

Amass

Enhance Amass with Data Sources

$ amass enum -list

Note amass can be used as a passive or active tool with the -passive or -active option.

Scan your target

$ amass enum -active -d target-name.com |grep api

Gobuster

$ gobuster dir -u target-name.com:port -w /home/ hapihacker/api/wordlists/common_apis_160

=============================================== ===========

[+] Url: http://192.168.195.132:8000

[+] Method: GET

[+] Threads: 10

[+] Wordlist: /home/hapihacker/api/wordlists/ common_apis_160

[+] Negative Status codes: 404

[+] User Agent: gobuster

[+] Timeout: 10s

=============================================== ===========

/api (Status: 200) [Size: 253]

/admin (Status: 500) [Size: 1179]

/admins (Status: 500) [Size: 1179]

/login (Status: 200) [Size: 2833]

/register (Status: 200) [Size: 2846]

Kiterunner

$ kr scan -w ~/api/wordlists/data/kiterunner/routes- large.kite

ALL THE API WORDLISTS

https://wordlists.assetnote.io/

Analyze Endpoints

Now that you've discovered an API, what can you do with it?

- Use the API as intended
- Reverse Engineer docs if necessary
- Check for Business Logic Flaws and Excessive Data Exposure

Reverse Engineering APIs

Multiple ways to reverse engineer an API:

- Creat Postman requests by hand
- Proxy with Cleanup
- Mitmproxy2swagger

Creating requests by hand can be time consuming, but can let you customize your requests. Better than nothing!

Proxy web traffic with FoxyProxy over to Postman

MitmProxy2Swagger!

$ mitmweb

In the web browser proxy all your traffic to port 8080.

METICULOUSLY use the web app.

$sudo mitmproxy2swagger -i /Downloads/flows -o spec.yml -p http://crapi.apisec.ai -f flow

Edit the specs to include all the endpoints that you want to target:

$sudo nano spec.yml

1

2

Once again:

$sudo mitmproxy2swagger -i /Downloads/flows -o spec.yml -p http://crapi.apisec.ai -f flow

You can check out your results in the Swagger Editor:

https://editor.swagger.io

Import the spec into a Postman Collection

Now you have a full collection of all the API requests that you can use to test out your target!

Review Docs and Use the API as Intended

Review API Docs

- What's required to make successful requests
- Get an idea of business logic
- Seek out interesting requests

MAKE API REQUESTS

- Use the API as an end user
- Get a lay of the land
- How do you authenticate? What is used for authorization? How does the API provider respond to failed vs successful requests?

Excessive Data Exposure

As early as this stage you could find critical vulnerabilities like Excessive Data Exposure.

Look for requests that supply more than you requested.

Check for information disclosure.

API3: Excessive Data Exposure

Excessive data exposure occurs when a consumer makes an API request for data and the provider responds with more information than requested

Excessive data exposure typically takes place because the API developers are expecting/trusting their consumers to parse out the data.

This vulnerability is the equivalent of asking someone for their name and they start telling you their DOB, home address, email address, SSN, and whether they use MFA

Example:

What you see in the web browser is not always what is sent by the API.

Behind the Scenes: Proxy the requests used for the Community Forum

The Response: Excessive Data Exposure!

Authorization Testing

Three Ingredients

- Resource ID
- Request Involving Resources
- Vulnerable API Provider

Look through docs/collection for relevant requests

Seek out responses that provide you with resource IDs

Make requests as UserA for UserB's resources

BOLA Cheat Sheet

crAPI Functionality

"When time has come to buy your first car, sign up for an account and start your journey."

http://crapi.apisec.ai:8025

BOLA!

Apply Your Skills

crAPI: https://github.com/OWASP/crAPI

vAPI: https://github.com/roottusk/vapi

vAmPI: https://github.com/erev0s/VAmPI

Pixi: https://github.com/DevSlop/Pixi

Additional Resources

- Awesome API Security: https://github.com/arainho/ awesome-api-security
- OWASP API Security Project: https://owasp.org/ www-project-api-security/
- Subscribe to apisecurity.io!
- Hacking APIs
- Follow these awesome API hackers:
    - @alissaknight
    - @InsiderPhD
    - @InonShkedy
    - @dolevfarhi
    - @theXSSrat

FREE Hacking APIs Companion Course: apisecu.com

hAPI Hacking!

FREE Hacking APIs Companion Course:

APISecUniversity: apisecu.com

corey.ball@mossadams.com

@hAPI_hacker

www.linkedin.com/in/coreyball

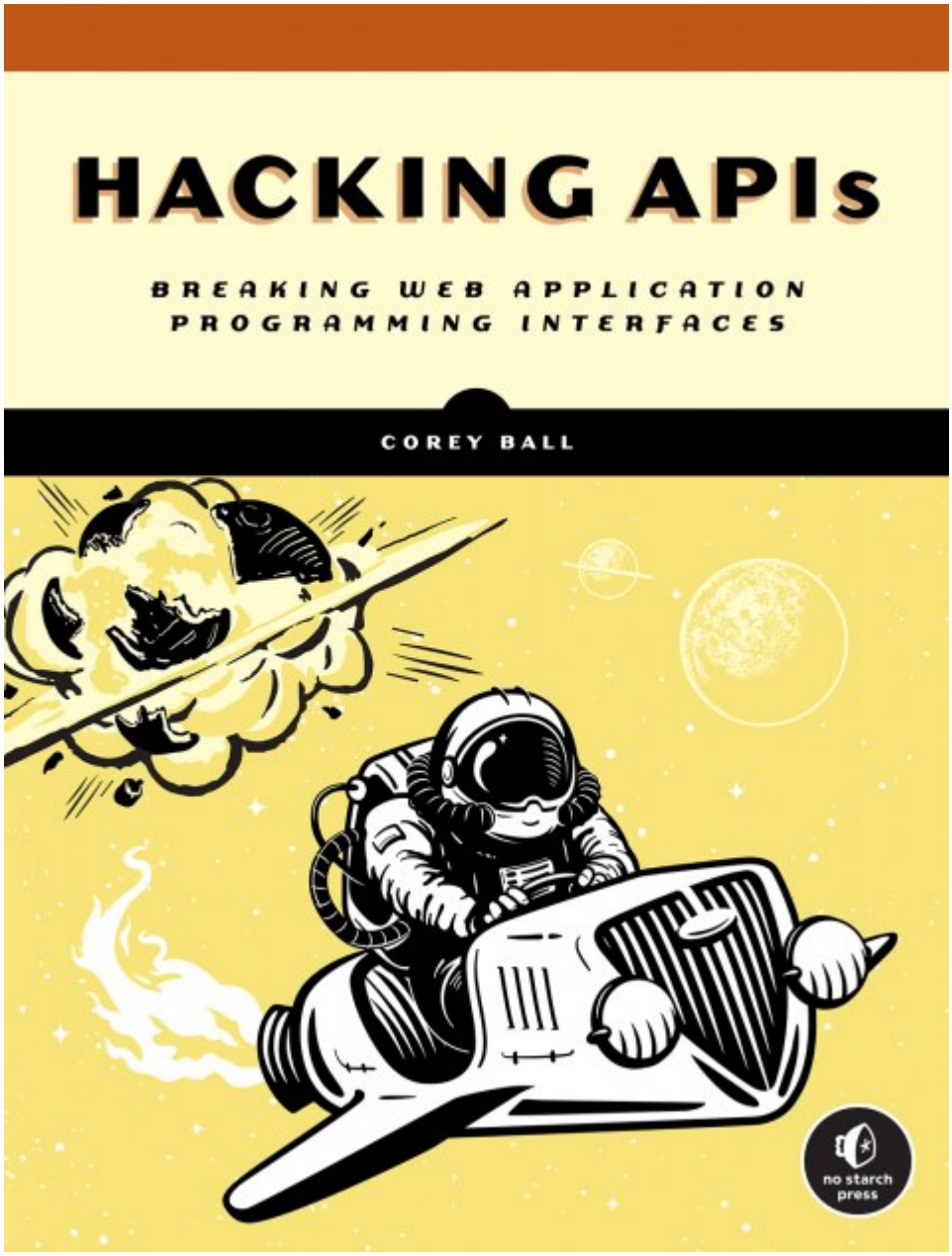
Made with Microsoft Sway

Create and share interactive reports, presentations, personal stories, and more.

Hacking APIs: Workshop

01

#whoami

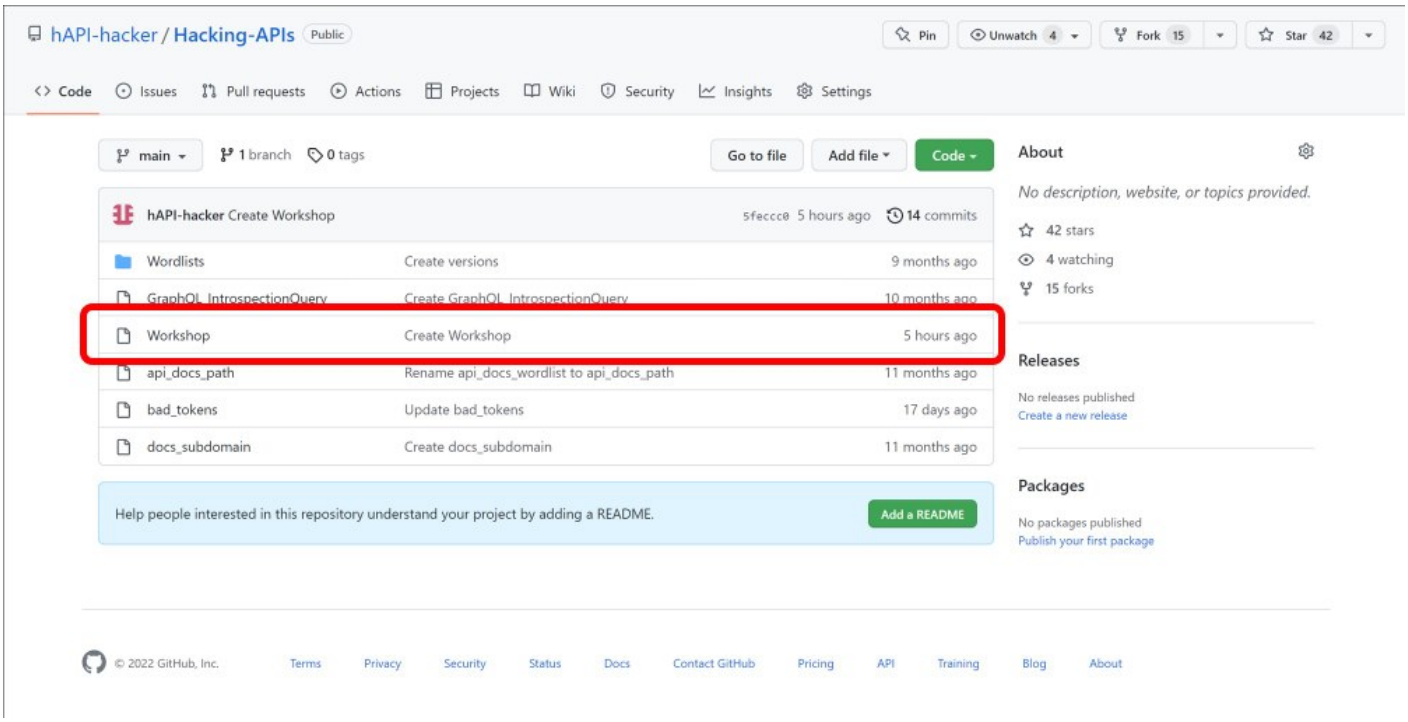



02

Preparing Your API Hacking System

03

https://bit.ly/3QEu9me




04

Goals

Find APIs

05

What are APIs?

06

API Vulnerabilities for Breakers

07

Discover APIs

08

Active Recon

09

Analyze Endpoints

10

MitmProxy2Swagger!

```
┌──(hapihacker㉿HackingAPIs)-[~]
└─$ mitmweb
Web server listening at http://127.0.0.1:8081/
Proxy server listening at *:8080
127.0.0.1:35116: client connect
127.0.0.1:35116: server connect crapi.apisec.ai:80 (20.230.217.15:80)
127.0.0.1:35120: client connect
127.0.0.1:35122: client connect
127.0.0.1:35124: client connect
127.0.0.1:35124: server connect crapi.apisec.ai:80 (20.230.217.15:80)
127.0.0.1:35120: server connect crapi.apisec.ai:80 (20.230.217.15:80)
127.0.0.1:35122: server connect crapi.apisec.ai:80 (20.230.217.15:80)
```

11

API3: Excessive Data Exposure

12

Authorization Testing

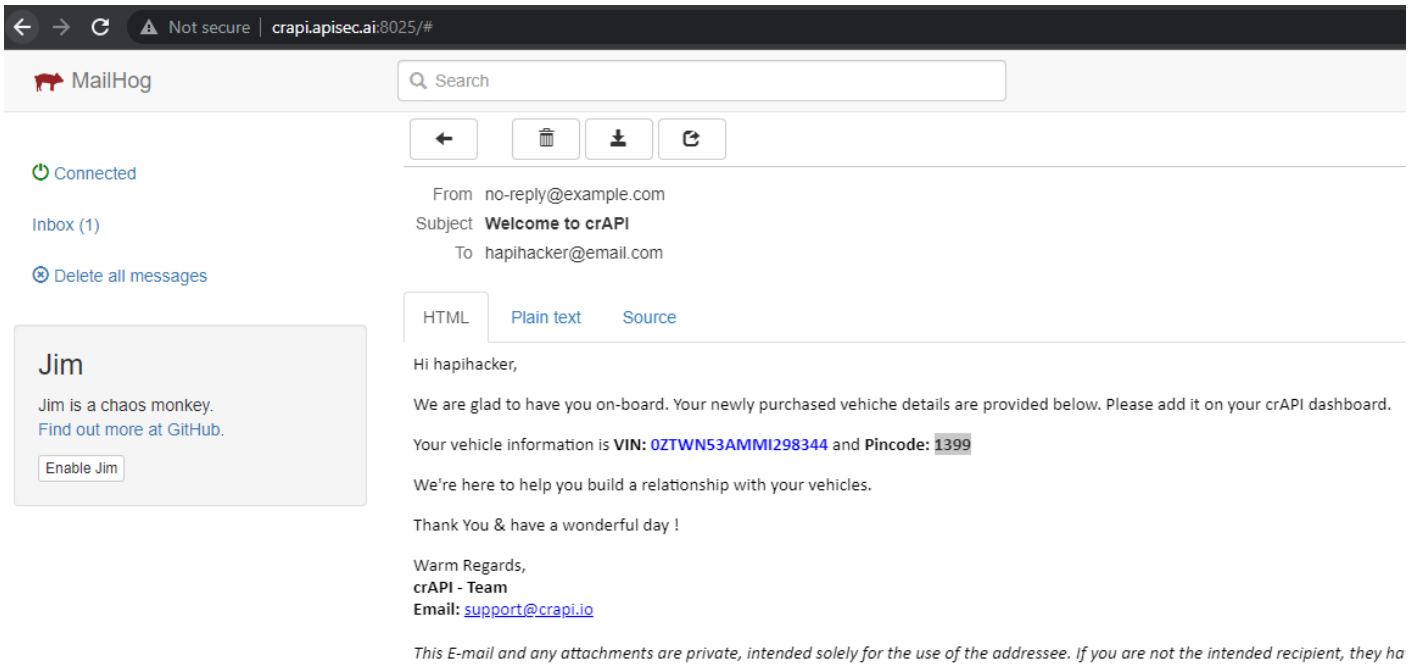
13

BOLA Cheat Sheet

| Type | Valid Request | BOLA Test |
|---|---|---|
| **Predictable ID** | GET /api/v1/account/**2222**<br>Token: UserA_token | GET /api/v1/account/**3333**<br>Token: UserA_token |
| **ID Combo** | GET /api/v1/**UserA**/data/**2222**<br>Token: UserA_token | GET /api/v1/**UserB**/data/**3333**<br>Token: UserA_token |
| **Integer as ID** | POST /api/v1/account/<br>Token: UserA_token<br>{"Account": **2222**} | POST /api/v1/account/<br>Token: UserA_token<br>{"Account": [**3333**]} |
| **Email as UserID** | POST /api/v1/user/account<br>Token: UserA_token<br>{"email": "**UserA@email.com**"} | POST /api/v1/user/account<br>Token: UserA_token<br>{"email": "**UserB@email.com**"} |
| **GroupID** | GET /api/v1/group/**CompanyA**<br>Token: UserA_token | GET /api/v1/group/**CompanyB**<br>Token: UserA_token |
| **Group and User Combo** | POST /api/v1/group/**CompanyA**<br>Token: UserA_token<br>{"email": "**userA@CompanyA.com**"} | POST /api/v1/group/**CompanyB**<br>Token: UserA_token<br>{"email": "**userB@CompanyB.com**"} |

14

crAPI Functionality




15

Apply Your Skills

16

Additional Resources

Awesome API Security: https://github.com/arainho/a

17

FREE Hacking APIs Companion Course: apisecu.com


18

hAPI Hacking!

FREE Hacking APIs Companion Course: