RESTful API Discovery

Ethical Hacking of RESTful & GraphQL API Training Course

REST API Discovery

- In Penetration Testing or Bug Bounty Hunting the easiest way to discover API endpoint is to walk the app (unauthenticated vs. authenticated)
- Visit every page, every function, click any button, perform any action possible while proxying through Burp
- Use Burp Scanner & Content Discovery
- Backslash Powered Scanner BApp to identify server-side injection vulnerabilities.
- Find out as much about the API as possible

REST API Discovery

- On any endpoint you find, try all possible CRUD (Create, Read, Update, Delete) Methods
- GET, POST, PATCH, PUT, DELETE, TRACE, OPTIONS
- Be careful to only test with your own accounts (especially when using the DELETE Method)
- Often OWASP vulnerability exists in one method but not in others

REST API Discovery (versions)

- You will often see a version number on API queries

  for example /api/v2.0/ or /api/getuser?v=2.0
- Try older versions such as v1.0

REST API Discovery – (content-type)

- Try a different content-type
- If you see application/json

  Send the request with application/xml
- and vice versa!
- Might support XML input -> XXE vulnerabilities?
- Use Content type converter extension from BApp

REST API Discovery – OSINT

OSINT examples (Google dorking)

inurl: /api/
intitle:"<targetname> api key"
inurl:"/wp-json/wp/v2/users"
intitle:"index.of" intext:"api.txt"
inurl:"/includes/api/"
intext:"index of /"
ext:php inurl:"api.php?action="
intitle:"index of" api_key
intitle:"index of" api_key OR "api key" OR apiKey –pool

REST API Discovery – OSINT

OSINT examples (Public API websites)

https://www.programmableweb.com
https://rapidapi.com
https://apis.guru

REST API Discovery – OSINT

OSINT examples (GitHub)

Search in code:

"api-key," "password," or "token"

(such as "API," "key," and "secret")

"Company" stage (staging, stg, dev, prod, qa, swagger)

"Company" apiKey (apiSecret, x-api-key, apidocs, /api/, internal/api)

Issues tab!

Pull Requests!

REST API Discovery – JavaScript

JavaScript

Go through all JavaScript files on the application you are testing (files end in .js)

In there search for common terms like:
“/api/”, “api” “v1”, “v2”, “v3”, “swagger”, “dev”, “rest”, “https://” ”http://”,

REST API Discovery – JavaScript

JavaScript (cont.)

api
api/
internal
url:
var =
//
/*
*/
http://
https://
parameter

REST API Discovery – JavaScript

JavaScript (cont.)

POST
GET
setRequestHeader
send(
.headers
.theirdomain.com
apiKey
location.href
redirectUrl
company.com
location.search

REST API Discovery - Documentation

Documentation

Often companies provide API documentation for 3rd parties to integrate.

https://example.com/docs
https://example.com/api/docs
https://docs.example.com
https://dev.example.com/docs
https://developer.example.com/docs
https://api.example.com/docs
https://example.com/developers/documentation

https://developer.twitter.com/en/docs/twitter-api

REST API Discovery - Documentation

Documentation

Use Burp scanner to crawl the API

Manually look for endpoints:
/api
/swagger/index.html
/openapi.json

If you identify /api/swagger/v1/users/123
Then investigate:
/api/swagger/v1
/api/swagger
/api/

REST API Discovery - Wordlists

Wordlists

https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content/api
https://github.com/chrislockard/api_wordlist
https://wordlists.assetnote.io
https://wordlists-cdn.assetnote.io/data/automated/httparchive_apiroutes_2023_08_28.txt
https://wordlists-cdn.assetnote.io/data/kiterunner/routes-large.kite.tar.gz
https://wordlists-cdn.assetnote.io/data/kiterunner/routes-small.kite.tar.gz

https://github.com/danielmiessler/SecLists
https://github.com/fuzzdb-project/fuzzdb

REST API Discovery - Kiterunner

Kiterunner

https://github.com/assetnote/kiterunner

curl https://wordlists.cdn.assetnote.io/data/automated/httparchive_apiroutes_2021_06_28.txt > latest_api_wordlist.txt

kr scan http://127.0.0.1:8000/ -w ~/Downloads/routes-large.kite

kr brute http://192.168.1.2:8000/ -A=apireoutes-210228 (requires plain wordlist - not kite)

Authenticated Scan:
kr scan http://192.168.50.35:8090 -w ~/api/wordlists/data/kiterunner/routes-large.kite -H 'x-access-token: TOKEN'

REST API Discovery - Wfuzz

https://github.com/xmendez/wfuzz

wfuzz -z file,/usr/share/wordlists/list.txt http://targetname.com/FUZZ

wfuzz -X POST -z list,admin-dashboard-docs-api-test
http://targetname.com/FUZZ

wfuzz -z range,500-1000 http://targetname.com/account?user_id=FUZZ

REST API Discovery - Gobuster

https://github.com/OJ/gobuster

gobuster dir -u http://192.168.195.132:8000 -w /home/hapihacker/api/wordlists/common_apis_160

gobuster dir -u http://targetaddress/ -w /usr/share/wordlists/api_list/common_apis_160 -x 200,202,301 -b 302

REST API Discovery – More Info

Resources

https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/web-api-pentesting

https://owasp.org/www-project-api-security/

API VMs

https://github.com/OWASP/crAPI

https://github.com/dolevf/Damn-Vulnerable-GraphQL-Application

Free Test APIs

https://reqres.in

https://swapi.tech

REST API Discovery – More Info

Arjun (Parameter discovery)
https://github.com/s0md3v/Arjun

Nikto (Web scanner)
https://github.com/sullo/nikto

Nmap (Portscanner, vulnerability scanner)
https://nmap.org

Amass (enumeration, domains, intel)
https://github.com/owasp-amass/amass

/robots.txt

Paraminer Burp extension…..and many more…

REST API Discovery – More Info

https://github.com/streaak/keyhacks

https://security.stackexchange.com/questions/248528/does-api-access-token-that-only-have-access-to-public-information-need-to-be-kep

https://github.com/pichik/pcon

https://blog.assetnote.io/2021/04/05/contextual-content-discovery/

https://beeceptor.com

REST API Discovery – Vulnerabilities

- OWASP Top 10 (same as with regular web apps)
- IDOR / Access Control
- CORS issues
- SQL injection / NoSQL injection
- XSS
- SSRF
- Information Leakage
- Weak Authentication / No authentication
- Encryption or the lack of
- XXE
- Rate Limiting
- Business Logic
- …

REST API Discovery – TIP

- Often APIs for Web Applications and Mobile Applications are run by different teams -> different vulnerabilities / protections!!!

REST API Discovery – Method

1. Recon
2. Tech bugs (RCE, SQLI, XXE, XSS etc.)
3. Logical bugs (IDOR, priv. escalation, info leak etc.)

REST API Discovery – Info Gathering

generate error messages

send data types it's not expecting (string, int, bool, array)

try all methods

send malformed JSON

REST API Discovery – Targeted scan

send to intruder

add position to scan

right click

select scan define insertion points

error message checks burp extension

REST API Discovery – RCE/SQLi

RCE:

SSTI

File uploads

XXE

Stored XSS

SQLi:

Regular SQLi

Blind SQLi

REST API Discovery – Logic

logical bugs:

3 accounts (User A Org 1, User B Org 1, User A Org 2

Org 1 and Org 2

User A Org 1 vs. User B Org 1 (user within orgs)

User A Org 1 vs. User A Org 2 (user across orgs)

use intruder

IDORs with UUID

Try all calls unauthenticated as well

REST API Discovery – Logic

priv escalation:

1 admin Org 1

1 admin Org 2

1 user Org 1

1 user Org 2

Try admin calls with user (less privileged) cookie

Try crossover as well

Autorize Burp ext

Try all calls unauthenticated as well

Thank You!

Ethical Hacking of RESTful & GraphQL API Training Course