{ API:WORLD™ }

OCT 24–26 Santa Clara, CA OCT 31–NOV 2 Live Online

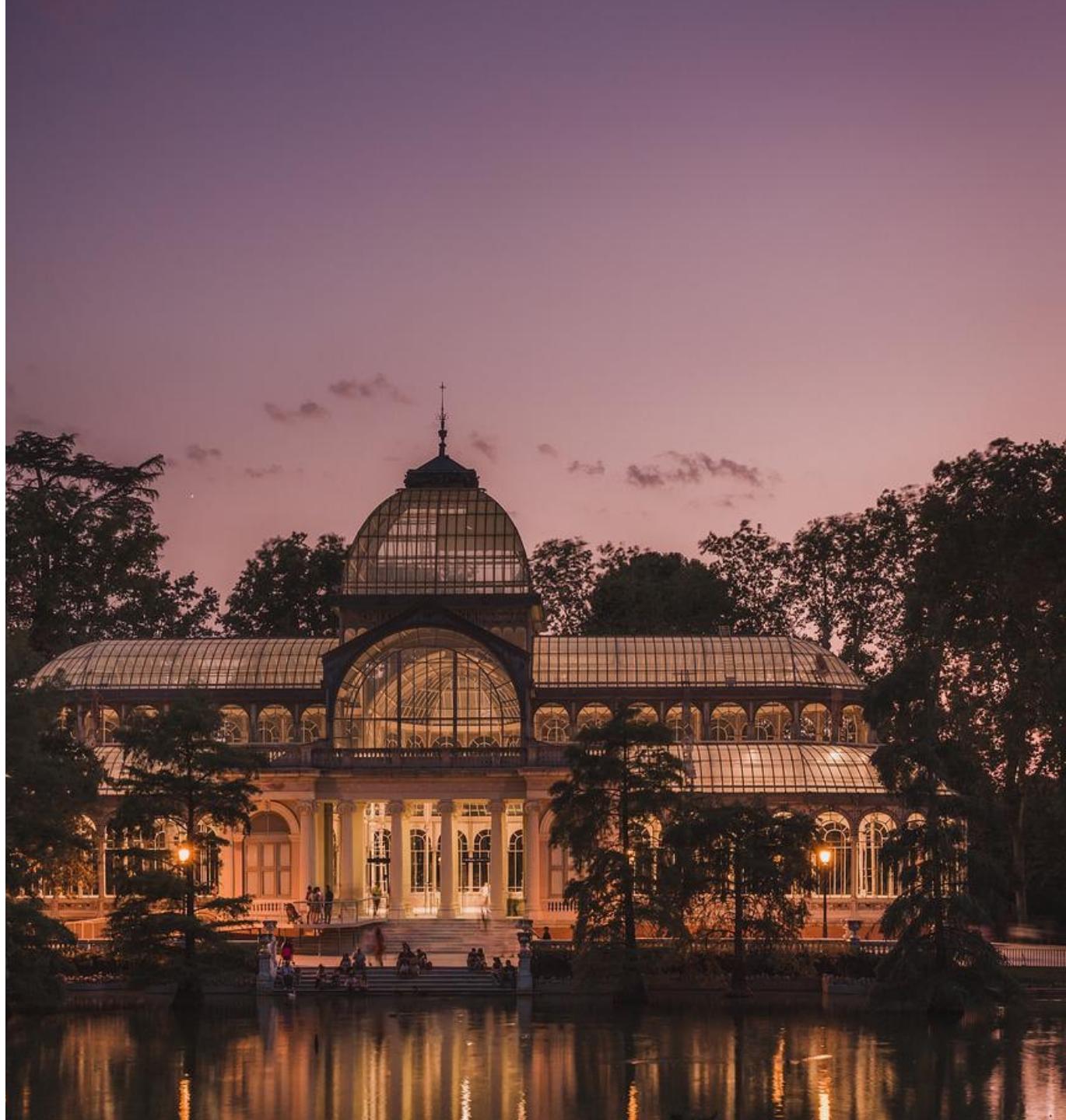
Common API Security Pitfalls

Isabelle MAUNY | @isamauny

Field CTO & Co-Founder - 42Crunch

Glad to be here!

- Field CTO / Founder of 42Crunch and apisecurity.io
- French National, I have lived in Spain for the past 20 years
- Most of career in the integration world, pioneering in 2004 what would become API Management
- This presentation and sample code at: https://github.com/isamauny/apiworld2023



APIs connect the world

Fintech

Stripe, Paypal , Google Pay are APIs

Healthcare

Huge impact during COVID time

IoT

Take control of your smart objects!

https://apisecurity.io

18 October, 23

Issue 231: API authentication bypass in Ivanti Sentry, Docker images expose API and keys

5 October, 23

Issue 230: OpenSea API breach, flaw in Atlas VPN, using API fuzzing

21 September, 23

Issue 229: Incidents with DuoLingo and JumpCloud, FastAPI for APIs, and five best practices

7 September, 23

Issue 228: 3rd party API security, OAuth2 step-up deep-dive, shadow and zombie APIs

25 August, 23

Issue 227: GhostToken on Google Cloud, Gartner on zero trust, API authentication

10 August, 23

Issue 226 : Jetpack WordPress plugin has API vulnerability, how to address API security in 2023

26 July, 23

Issue 225 : API security needs a reset, vAPI walkthrough, five stages to attain API security

6 July, 23

Issue 224 : API security is critical in 2023, API contract testing, and Fencer security testing tool

Nothing New !

Says the wise ChatGPT

when was the first input validation cheat sheet published by OWASP ?

The Open Web Application Security Project (OWASP) is an organization that focuses on improving the security of software. Their "cheat sheets" are concise collections of information about specific security topics, providing guidance on best practices.

As of my last update in January 2022, the OWASP Input Validation Cheat Sheet was first published in 2009. However, OWASP content can be updated or changed over time, so for the most up-to-date information or specific publication dates after 2022, I would recommend checking OWASP's official website or their repository directly.

WHY ARE APIS SUCH A PROBLEM?

APIs are the heart of application development and integration

"APIs are popping up like mushrooms in my data centers"

From a desperate Fortune 500 CISO

Security is still an afterthought!

Delivery pressure lead to untested APIs into the wild.

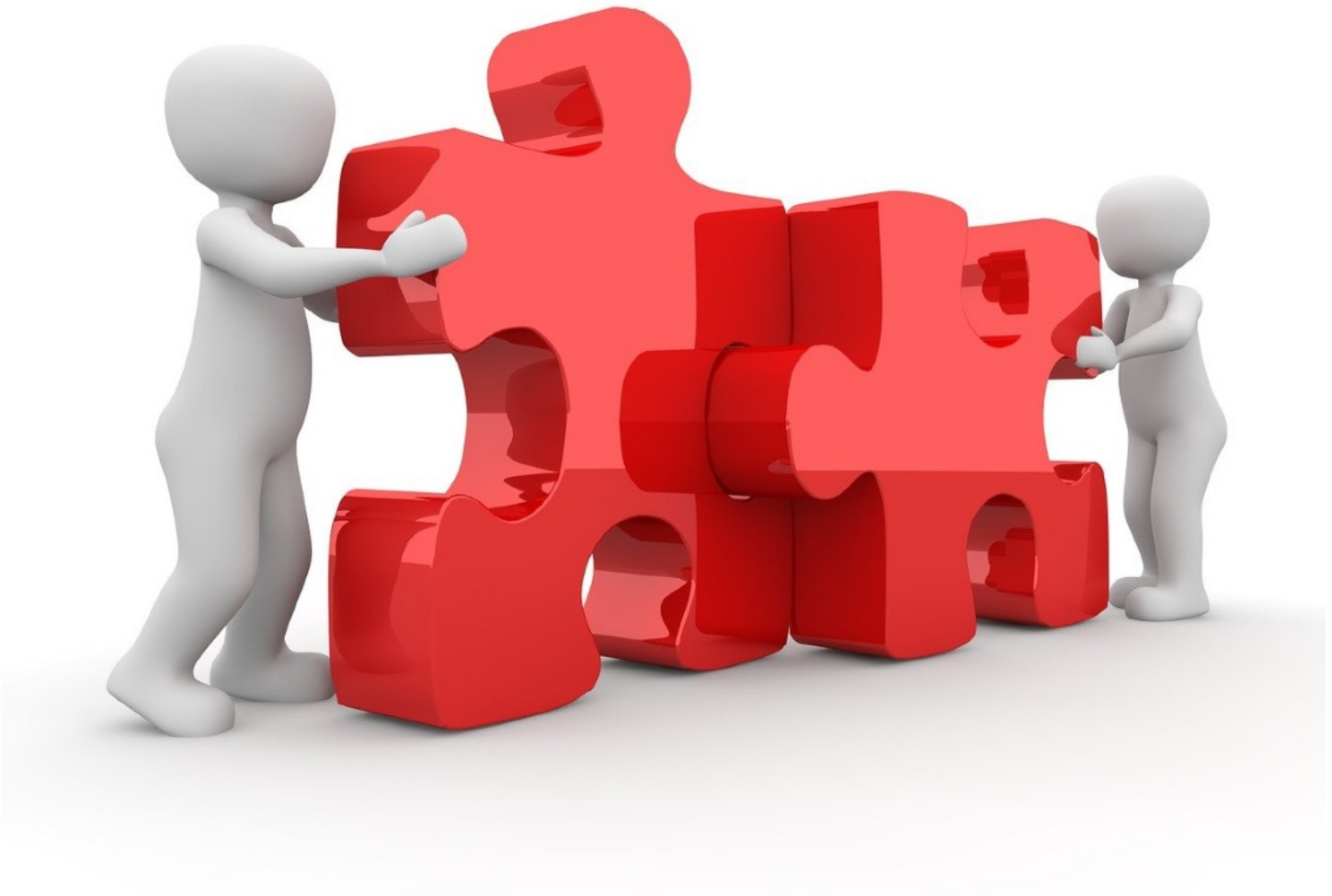
Design flaws are critical

APIs suffer from **many design flaws**, which are hard, sometimes impossible to fix without a redesign.

Humans Errors

Logic errors
Poor design
Coding errors
Misuse of components/libraries
Misconfiguration of servers
Shortcuts
Assumptions
Insecure defaults
Misunderstanding attack vectors
Vulnerable dependencies

LEARNING FROM API BREACHES

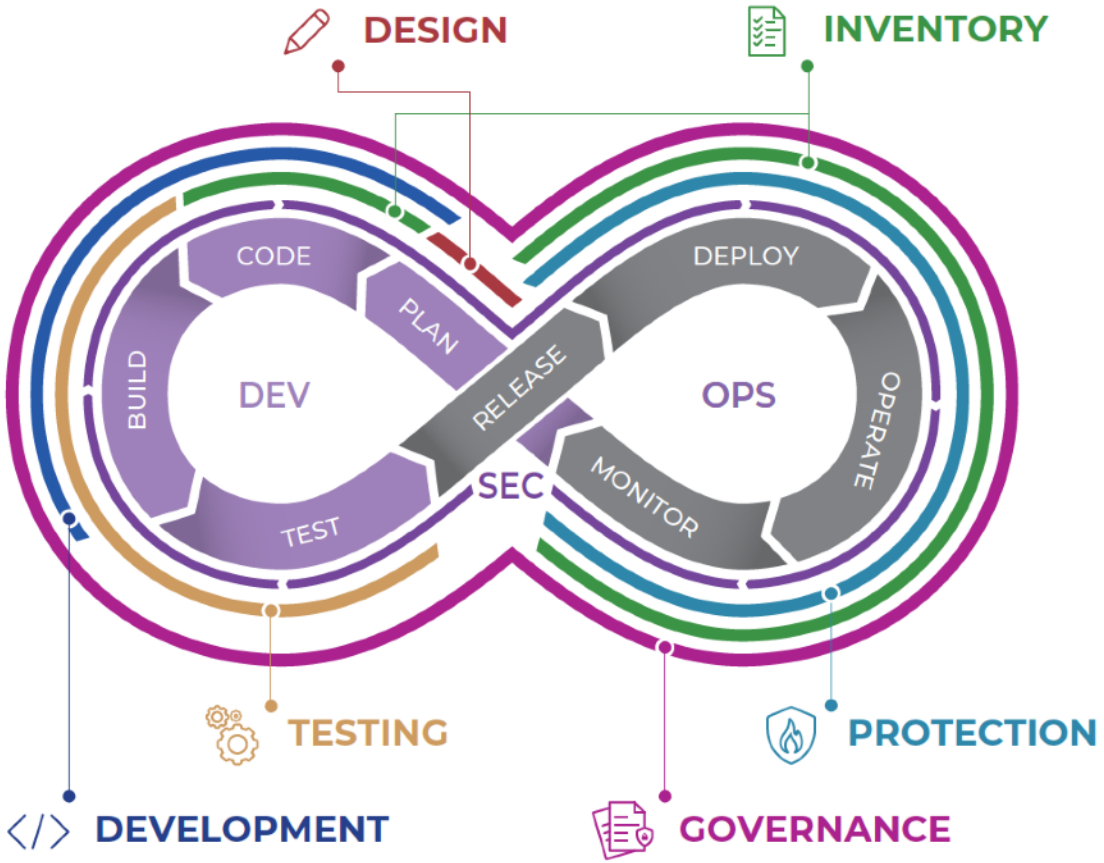


OWASP API Security Top 10 2023

- API1:2023 Broken Object Level Access Control a.k.a BOLA
- API2:2023 Broken Authentication
- API3:2023 Broken Object Property Level Authorization, a.k.a BOPLA (**Updated**)
- API4:2023 Unrestricted Resources Consumption
- API5:2023 Broken Function Level Authorization a.k.a BFLA
- API6:2023 Server Side Request Forgery (**New**)
- API7:2023 Security Misconfiguration
- API8:2023 Lack of Protection from Automated Threats (**New**)
- API9:2023 Improper Assets Management
- API10:2023 Unsafe Consumption of APIs (**New**)

BAD PROBLEMS USUALLY OCCUR WHEN MULTIPLE OF THESE ARE COMBINED

The six pillars of API security



API INVENTORY

Do you understand what APIs you own? Do you track shadow and zombie APIs?

API TESTING

Are you doing automated API testing? Are you considering security in your test strategy?

API DESIGN

Are you doing API-design-first? Do you incorporate security into the design phase?

API PROTECTION

Are you using API protection technology (WAFs, WAAPs, API gateways) in your deployments?

API DEVELOPMENT

Are your developers trained to code securely? Do they understand API security threats and risks?

API GOVERNANCE

Do you control and actively monitor your API estate and environments?

Get full eBook!

#1 Global shipping company

What happened?

- Researchers discovered they could automatically submit **parcel numbers** to an un-authenticated API that retrieved a map image.

- They then used this image and basic OSINT tools to guess the location from the image and obtain the **postcode**
- Now with parcel number and postcode, they have full access to the parcel data (and user information!)

Impact

- Potentially large-scale exfiltration of customer PII and parcel tracking information,
- Data could be leveraged through phishing campaign.

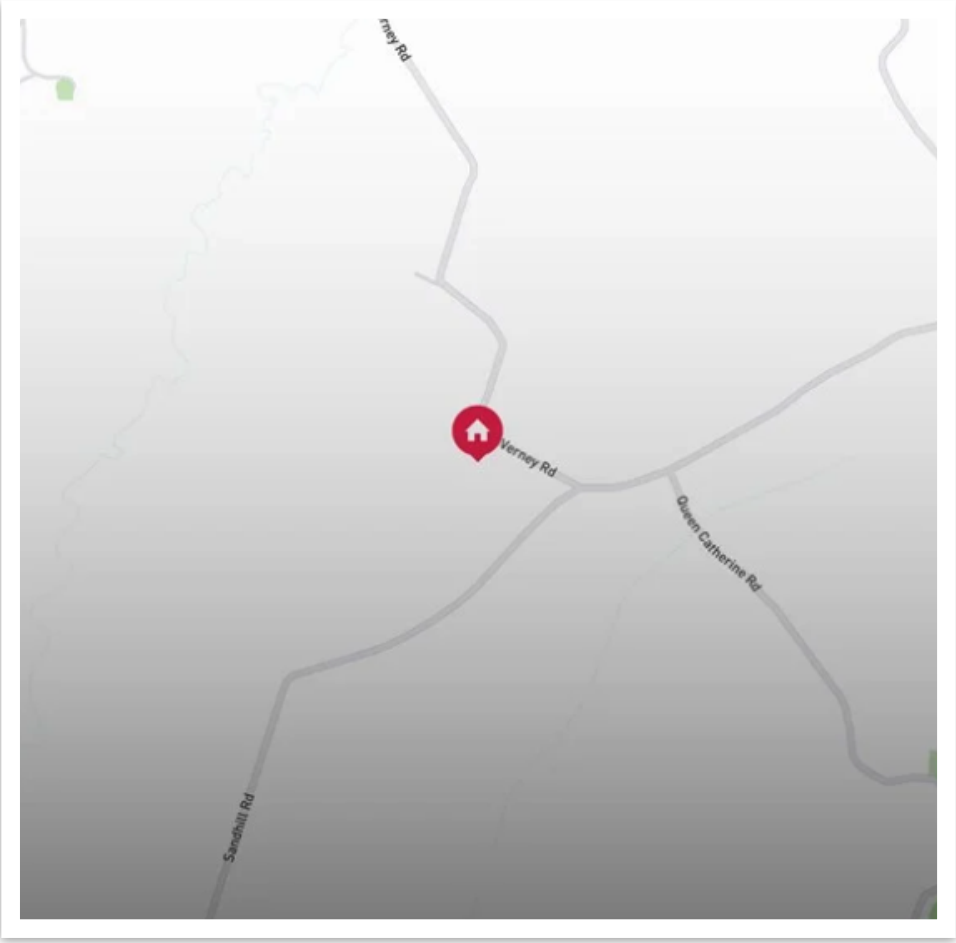
Root causes

- Lack of rate-limiting (API 4)
- Excessive information exposure (API 3)
- *No authentication on API (API 2)*

Resolution

- Researchers reported responsibly and a fix was released quickly.

#1: Global shipping company



JSON Raw Data Headers

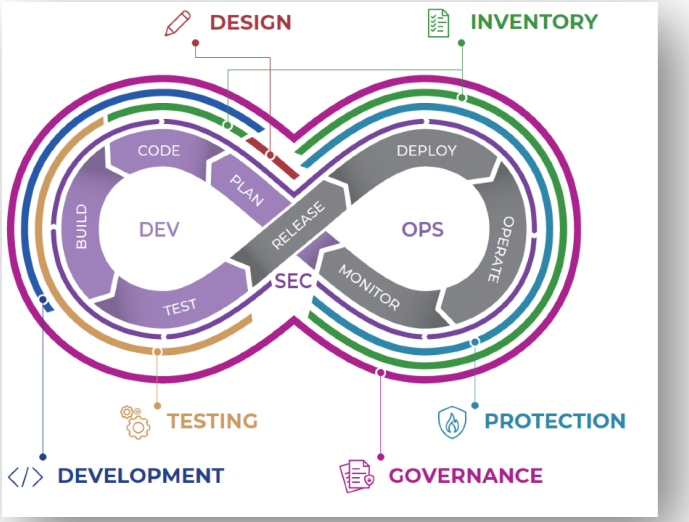
Save Copy Collapse All Expand All Filter JSON

```
addressPoint:
    longitude:          -0.93712
    latitude:           51.938844
notificationDetails:
    mobile:             [redacted]
    email:              null
    contactName:        "KEN MUNRO"
podDetails:
    podName:            "Tom"
```

#1: Global shipping company - how to prevent ?

Design

- Understand **PII requirements** and which data is returned by the API
- Understand **abuse case** - brute forcing/enumeration of parcel codes
- Extra caution at design time when no authentication is in place.



Testing

- Control APIs response under duress
- Data filtering must happen server side!

Protection

- Implement rate limiting on critical APIs
- In general, endpoints must have some type of authentication.

#2: Campus access control

What happened?

- Out of frustration with the campus app, a student went straight to the API to open campus doors
- The backend API that did not really authenticate users, allowing an attacker to impersonate any user given their guessable (public!) IDs.
- By faking the user location an attacker could access all doors on campus.
- They could also order food, view transactions history and obtain the email and phone number of any account

Root Causes

- Broken function-level authorization (API 5)
- Broken authentication (API 2)

#2: Campus access control - how to prevent

Design

- If geolocation is used, make sure it can't be overridden easily
- Design non guessable identifiers

Development

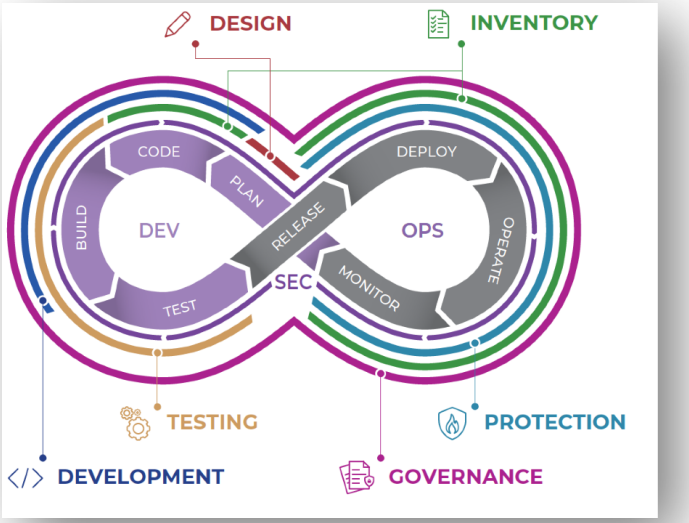
- Implement proper authorisation mechanisms

Testing

- Test for broken authentication and authorisation

Governance

- Getting the problem recognised by the company was very hard: implement responsible security disclosure



#3: Microbrewery application

What happened?

Mobile application for microbrewery used hardcoded tokens within application binary which could easily be extracted allowing for manipulation of backend functions including other users PII, and access to discount schemes, etc.

Impact:

Free beer !! Disclosure of user's PII.

Root Cause:

- Hardcoded tokens in mobile application (API 7)

Lessons learned:

- Mobile apps can be reverse engineered!
- Use a standard mechanism (such as OAuth2) for the exchange and distribution of tokens.
- Favour short lived tokens

```
getUser:function(t){return
o.default.get("https://www.brewdog.com/uk/rest/uk/V1/customers/"+t,{headers:{'Cache-
Control':'no-cache, no-store, must-revalidate',Pragma:'no-cache',Expires:0,Authorization:"bearer
y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})},

getUserWithUsername:function(t){return
o.default.get("https://www.brewdog.com/uk/rest/uk/V1/customers/search?searchCriteria[filterGro
ups][0][filters][0][field]=email&searchCriteria[filterGroups][0][filters][0][value]="+t+"&searchCriteria
[filterGroups][0][filters][0][conditionType]=equals",{headers:{'Cache-Control':'no-cache, no-store,
must-revalidate',Pragma:'no-cache',Expires:0,Authorization:"bearer
y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})},

setMyLocal:function(t,s,n){return
o.default.put("https://www.brewdog.com/uk/rest/uk/V1/customers/"+t.id,{customer:{id:t.id,group
_id:t.group_id,email:t.email,firstname:t.firstname,lastname:t.lastname,store_id:t.store_id,website_i
d:t.website_id,custom_attributes:[{attribute_code:'my_local_id',value:s},{attribute_code:'my_local_
reset_date',value:n}]}},{headers:{Authorization:"bearer y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})};
```

#3: Microbrewery application - how to prevent it

Design

- Should be using standard authorization framework
- PII requirements design

Testing

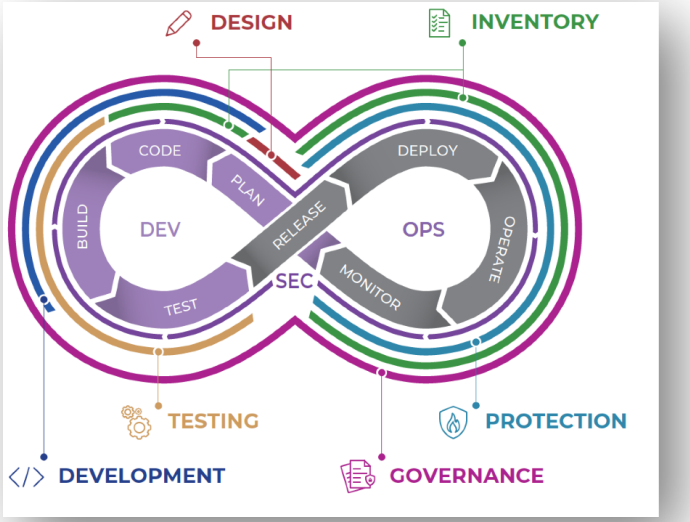
- Hardcoded tokens should have been detected

Protection

- No monitoring – no way to know if users actually abused the vulnerability

Governance:

- Implement security process to handle security disclosure
- Make addressing issues a priority



#4: Cryptocurrency portal

What happened?

A researcher discovered an issue in a cryptocurrency trading platform whereby he could trade between two different accounts. The platforms failed to validate the account details and allowed purchases from accounts with insufficient funds. The exploit could be triggered by manipulating API request parameters.

Impact

Very limited due to responsible disclosure and immediate response.

Cause:

- A text-book case of broken-object level authorization (BOLA) allowing manipulation via an API parameter (API1)

Lessons learned:

- Broken object-level authorization is the number one API security issue — always ensure you fully validate access to objects for all requests.
- Bug bounties can be profitable — this was worth $250,000.

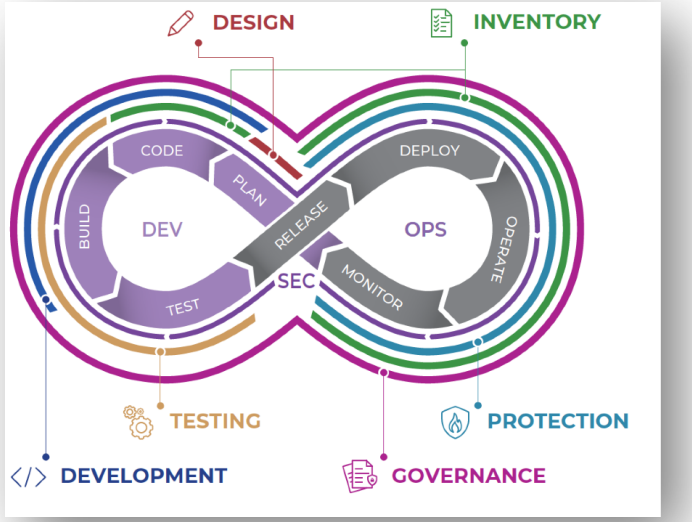
#4: Cryptocurrency portal - how to prevent it

Development

- Security training to raise awareness of BOLA
- Code reviews could have identified issue
- Design data access controls

Testing

- Testing for un-authorized access to a resource must be part of standard API testing



Protection

- Implement systematic/in depth data access authorisation validation (in code, in API Gateway/Firewall layer)

#5: Smart scale

What happened?

Researchers discovered that they could perform a variety of attacks on an API backend for a smart scale, including gaining access to access and refresh tokens, and account takeover using a ‘password reset’ functionality.

Impact:

Unknown, but vulnerabilities were remediated seven months after disclosure…

Cause:

- Broken authentication (API 2)
- Broken object-level authorization (API 1)
- Excessive data exposure/ BOPLA (API 3)

Lessons learned:

- Multiple vulnerabilities can be effectively combined to achieve total compromise.

#5: Smart scale - how to prevent it

Design:

- Insecure defaults – disable debug interface by default
- Threat modelling would have identified issues – guessable IDs, and PINs

Development:

- Security awareness training would prevent issues seen here

Protection:

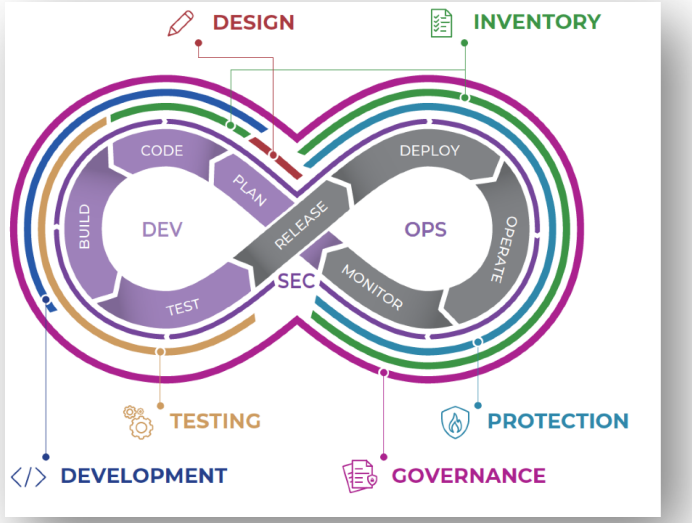
- Excessive information disclosure can easily be prevented by systematically validating responses
- Implement smart rate-limiting to prevent abuse of password reset function

Testing:

- Apply thorough testing to any endpoint related to token management

Governance:

- Implement responsible disclosure program



Your most sensitive endpoints are **authentication and password reset** endpoints.

#6: Major AWS access keys Leak

What happened?

- The attacker gained access to a set of AWS access keys by accessing the AWS EC2 metadata service via a SSRF vulnerability.

Impact

- One of the largest data breaches from the past 5 years.

Cause

- SSRF (API 6) due to WAF misconfiguration

Lessons learned

- Cloud services like AWS or Azure expose metadata API via fixed IP (169.254.169.254) which needs special protection.

#6: SSRF Attack - How to prevent ?

Design

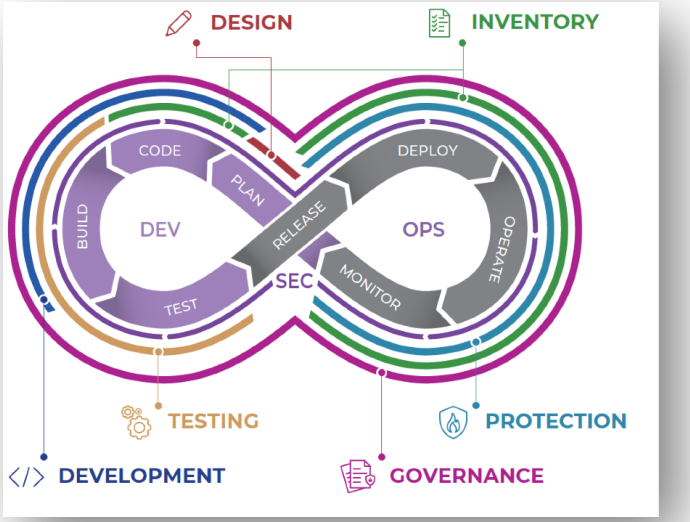
- Define list of authorised URLs the API needs access to.

Testing

- Thoroughly test APIs with bad URLs to detect SSRF issues

Protection

- Put in place an **allow** list of URLs, avoid Deny lists which are close to impossible to maintain
- For example, do you know what this pings : ping 2130706433 ?




BUILDING
RUGGED APIS

*Meaning of **rugged** in English*

rugged

adjective

UK /ˈrʌg.ɪd/ **US** /ˈrʌg.ɪd/

rugged ***adjective*** **(NOT EVEN)**

Add to word list

(of land) wild and not even; not easy to travel over:

- *rugged landscape/terrain/hills/cliffs*

+ SMART Vocabulary: related words and phrases

rugged ***adjective*** **(STRONG)**

strong and simple; not delicate:

- *Jeeps are rugged vehicles, designed for rough conditions.*

An API **must never blindly trust** anything it receives from the client.

And that includes

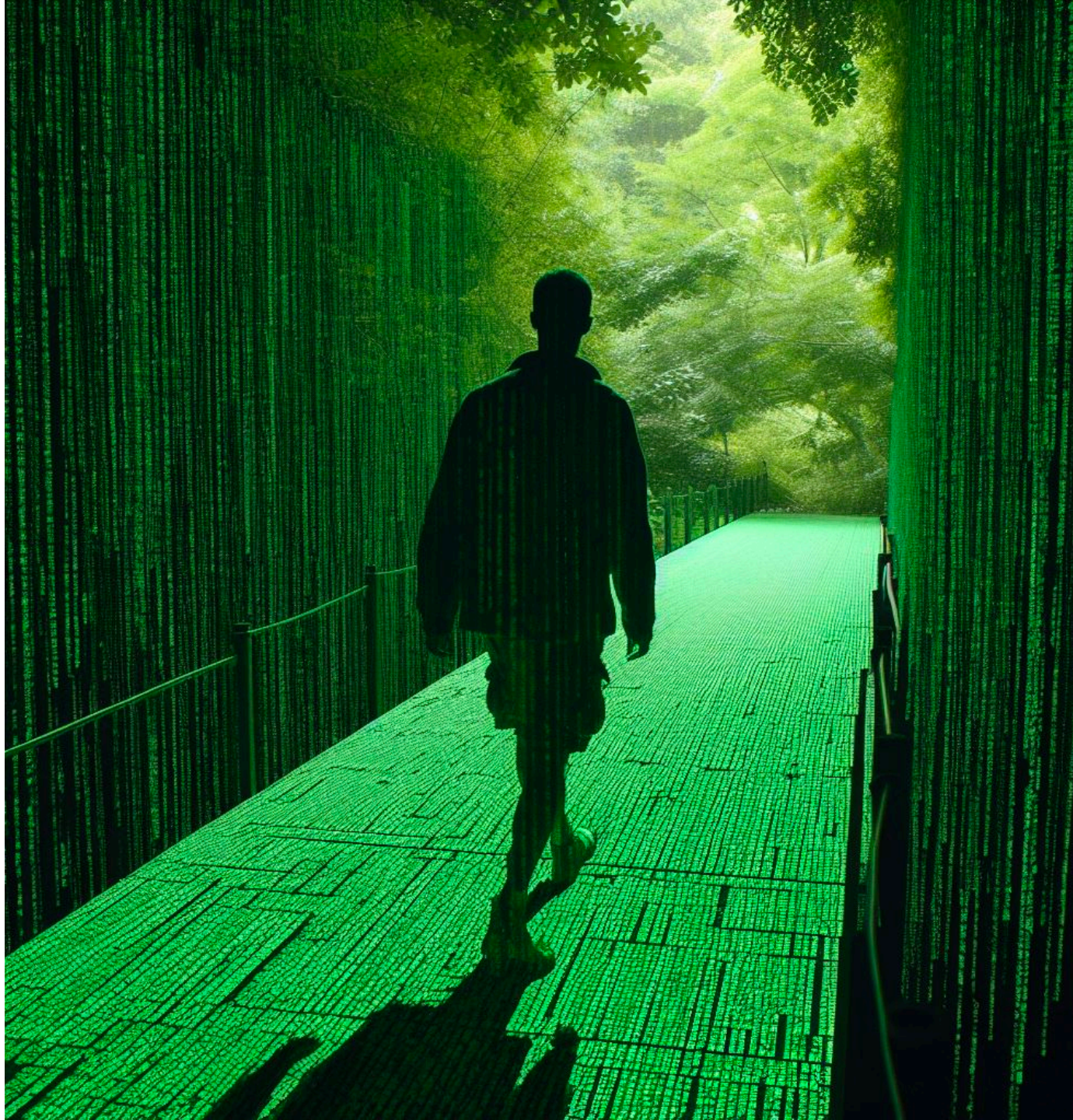
- Request payloads
- Headers
- JSON Web Tokens
- IP addresses (X-Forwarded-For)
- And everything else :)

Follow the “Hacky Path”

For **each happy path** test, you should have **10 Hacky Path** tests

And if you automate them, even better!

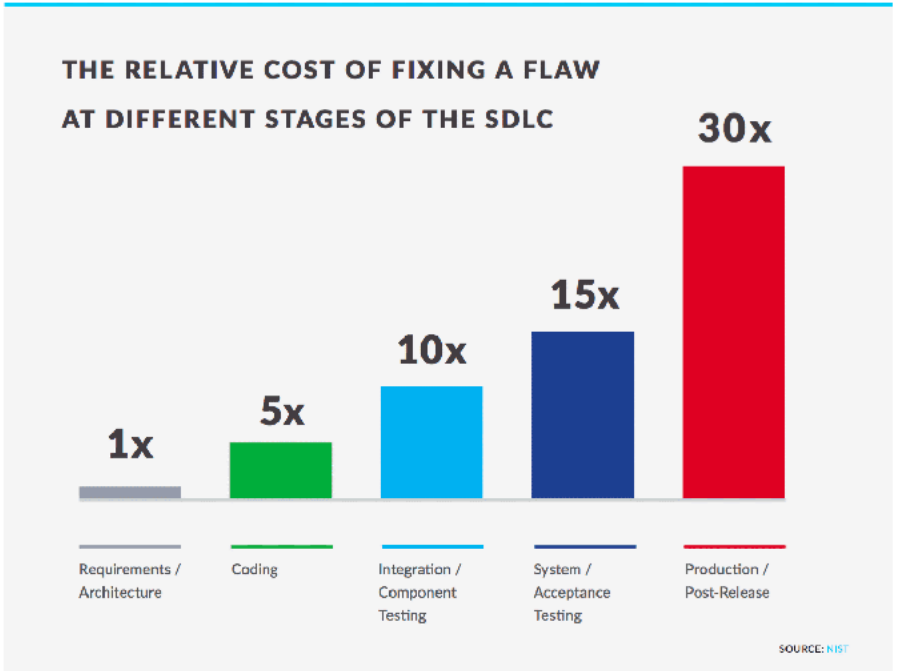
The way to protect yourself from Human Errors



CALL TO ACTION!

- Use OWASP API Top 10 as framework for design and testing
- Start with your critical APIs
- Start worrying about API Security at design time
  - ✓ A vulnerability discovered at production time costs up to 30x more to solve
- Hack yourselves leveraging API contracts
  - ✓ For each functional test, create 10 negative tests
  - ✓ Hammer your APIs with bad data, bad tokens, bad users
- Automate Security
  - ✓ Inject Security into DevOps practices and don't rely on manual testing of APIs.
  - ✓ Only solution to scale and have avoid human errors

https://www.helpnetsecurity.com/2020/05/20/devops-software-development-teams/



"I think security, in most cases, is not a single person's specialization. Security must be a practice of every member of the team from the frontend developer to the system administrator (also non tech roles)."

From: Gitlab DevSecOps report - 2021

There is more !

Wednesday, October 25 • 11:00am - 11:50am
- Deep Dive workshop on OWASP API Top Ten 2023 and how to proactively address those.

Thursday, November 2 • 9:30am - 9:55am
- Open Talk on “Why So Many API Security Solutions Have Failed to Deliver”

References

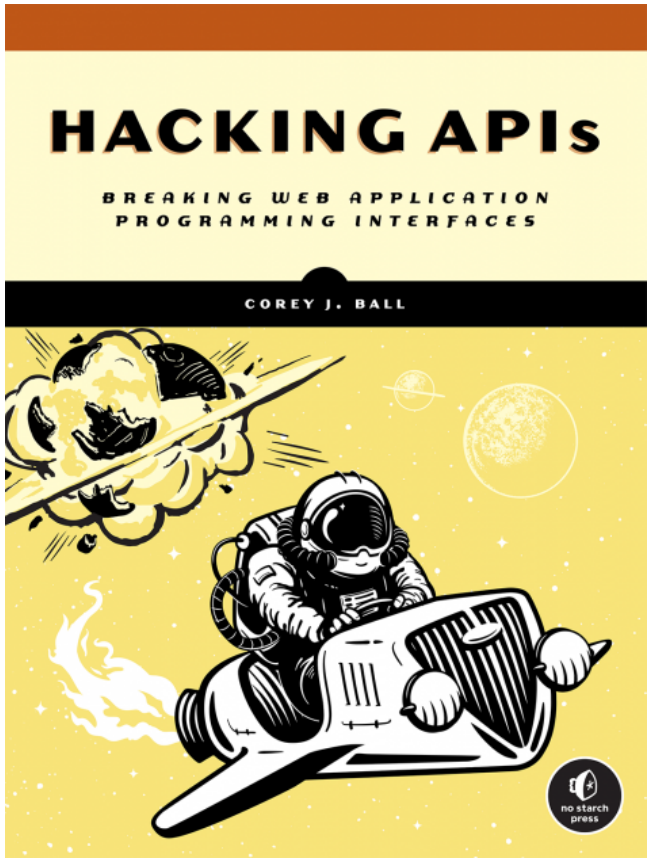
- **Evolution of OWASP API Top 10**: https://www.youtube.com/watch?v=ARIZNLzKwJI
- **BOLA 101**: https://inonst.medium.com/a-deep-dive-on-the-most-critical-api-vulnerability-bola-1342224ec3f2
- **OAuth Playground**: https://www.oauth.com/playground/
- **OAuth / OIDC Free Course**: https://pragmaticwebsecurity.com/courses/introduction-oauth-oidc.html
- **SSRF:**
  - **Intro @APIDays Paris**: https://www.youtube.com/watch?v=vG4n4ivsFnk
  - Overview and Labs: https://portswigger.net/web-security/ssrf
  - https://danaepp.com/exploiting-ssrf-in-an-api
    - And in general, Dana's blog!

Learning more

APISecurity.io

https://apisecurity.io/

“Hacking APIs” – Corey Ball

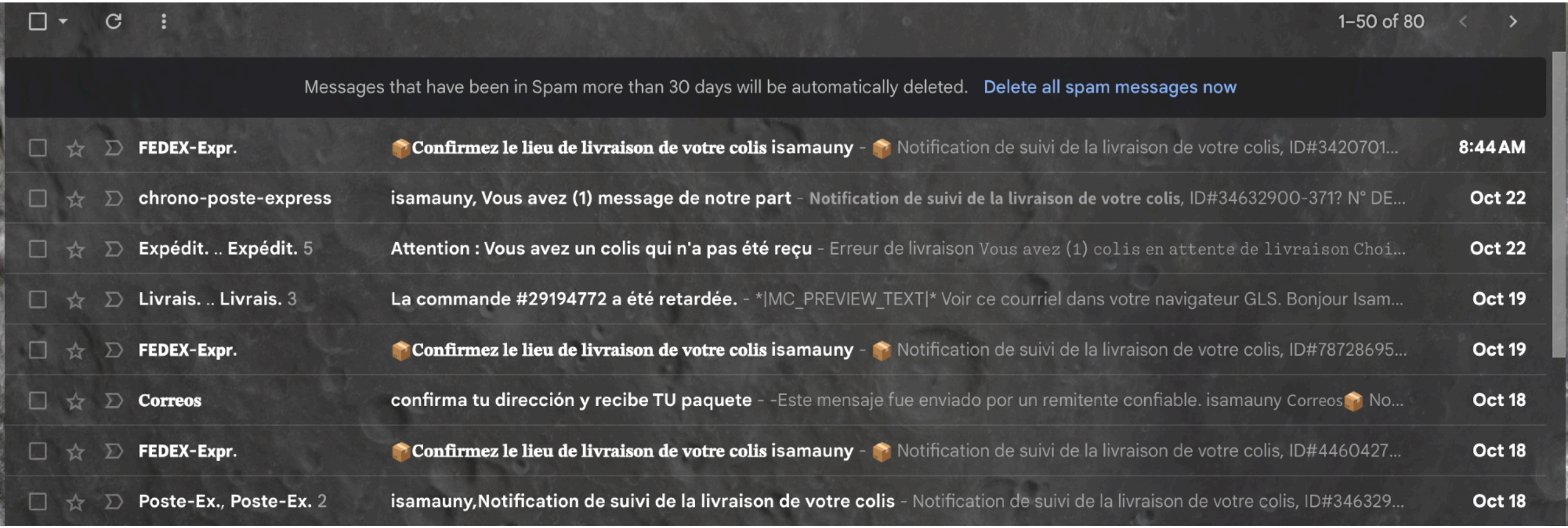


https://nostarch.com/hacking-apis

Learning Application Security

Buy the book

Last 3 days of parcel delivery SPAM!



Back to previous slide