Common API Breaches and how to avoid them

ISABELLE**MAUNY** - Field CTO

isamauny

Agenda

- Intro to Security vocabulary and principles
- OWASP API Security Top10
  - Data Validation
  - Authentication
  - Authorization
  - Rate Limiting
- Architecture and Tools
- General Recommendations

Why is API Security so hot right now?

- 400+ breaches reported on apisecurity.io since Oct. 2018
- And those are just the public ones!
- Recurring Combination of:
  - Lack of Input validation
  - Lack of Rate Limiting
  - Data/Exception leakage
  - Authorization issues
  - Authentication issues

https://siliconangle.com/2021/09/16/ibm-report-finds-two-thirds-cloud-breaches-traced-misconfigured-apis

Everyone is a target!

Facebook - 50 million users' personal information was exposed

PayPal - 1.6 million customers at risk of data exposure

T-Mobile - 76 million users' phone numbers and addresses stolen

Instagram - 49 million users' emails and phone numbers exposed

Uber - 57 million riders and drivers accounts were compromised

Justdial - Over 100 million Indian users' personal data at risk

Equifax - 147 million users personal data stolen

Starbucks - 100 million customer records accessed

Verizon - 14 million subscribers phone numbers and PINs exposed

Security Goals Overview

INTEGRITY	Message has not been tampered with
CONFIDENTIALITY	Message can only be seen by target audience
AVAILABILITY	Resistance to attacks, such as Denial-of-service (DOS)
AUTHENTICATION	Identity of the caller is known.
AUTHORIZATION	We can guarantee the caller has proper permissions to access a resource
AUDIT	System has non-perishable trace of all machine/human iteractions.
NON-REPUDIATION	There is (legal) proof that the action has taken place.


SECURITY
PRINCIPLES

API Security is risk based

- ☑ Threat Model
- ☑ Data Classification
- ☑ Actors Identification

You can’t protect what you don’t know!

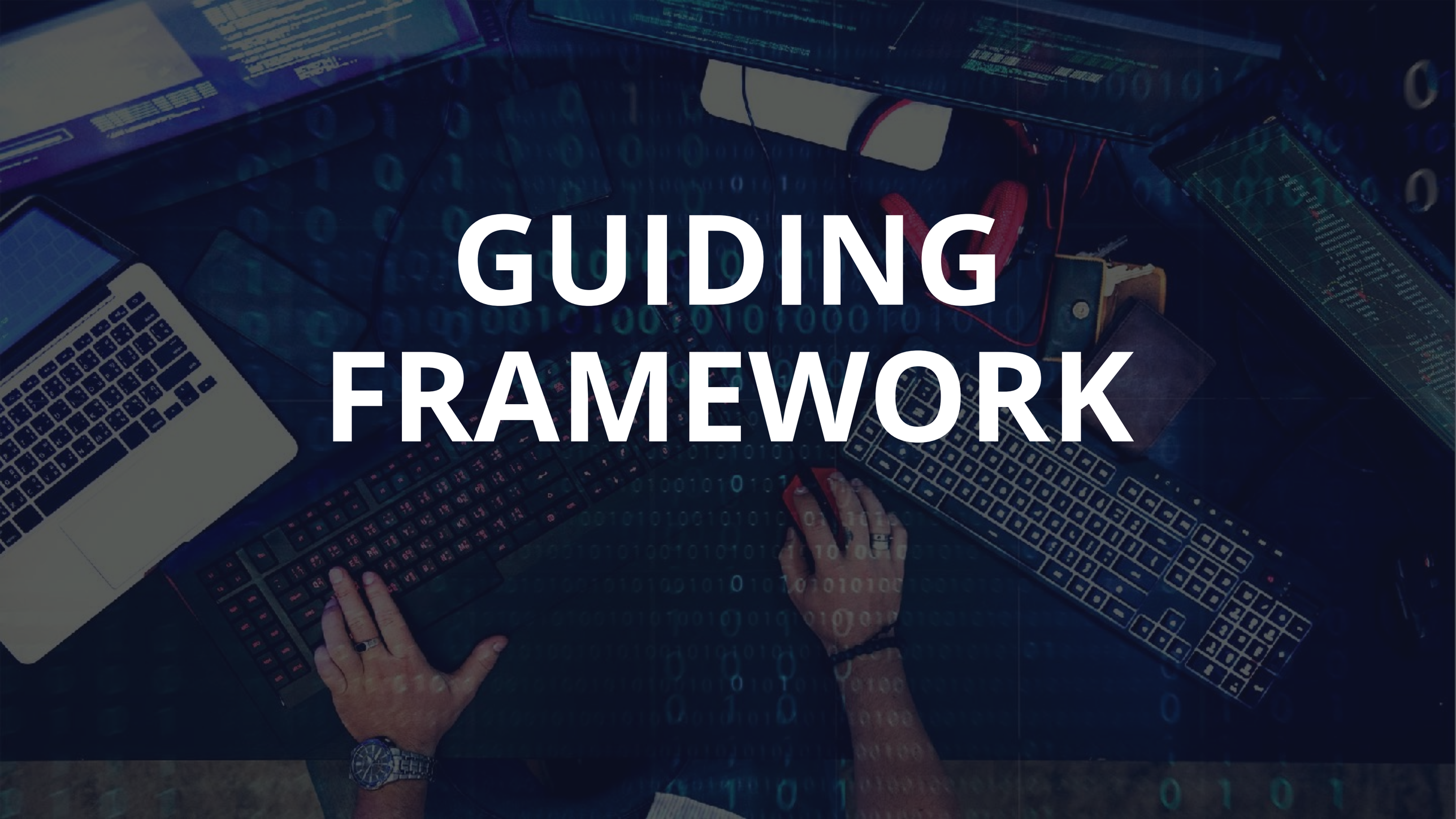
- ☑ API Catalog
- ☑ API Governance
- ☑ API Security Status

No Trust

THE GUIDING PRINCIPLE!

- Always question who/what is going to access the API
- Always question anything you reuse/adopt (anything from StackOverflow, libs, code)
- Always question any data you receive
- Also applies to internal traffic (especially in a service mesh)
- Apply Defense in Depth

“Treat APIs like they have a direct interface into your underlying systems and can bypass security controls – because that is pretty much what they do,” said Peter Liebert, former CISO, state of California


GUIDING
FRAMEWORK

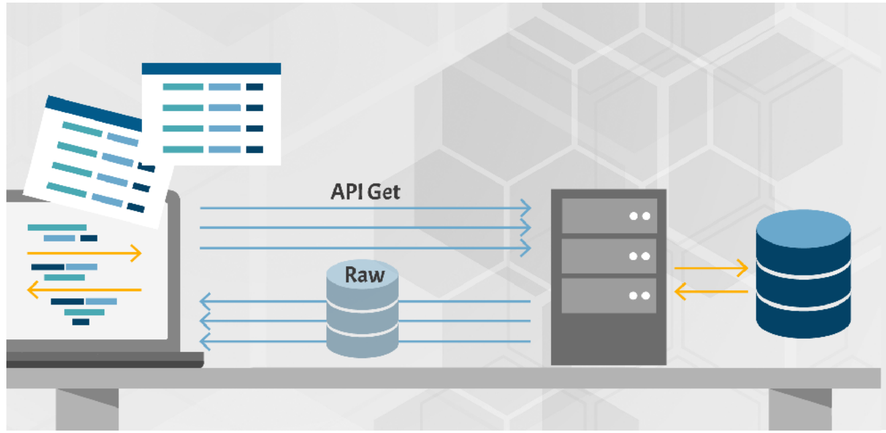
OWASP API Security Top 10

- API1 : Broken Object Level Access Control
- API2 : Broken Authentication
- API3 : Excessive Data Exposure
- API4 : Lack of Resources & Rate Limiting
- API5 : Missing Function Level Access Control
- API6 : Mass Assignment
- API7 : Security Misconfiguration
- API8 : Injection
- API9 : Improper Assets Management
- API10 : Insufficient Logging & Monitoring

Data Protection Auth / Authorization Governance/Operations


DATA
PROTECTION

Excessive Data Exposure (API 3)



Looking forward to generic implementations, developers tend to expose all object properties without considering their individual sensitivity, relying on clients to perform the data filtering before displaying it to the user.

FIZIKAL

Uber

tinder

Grindr

3Fun

הליכוד
Likud Party

jCrush

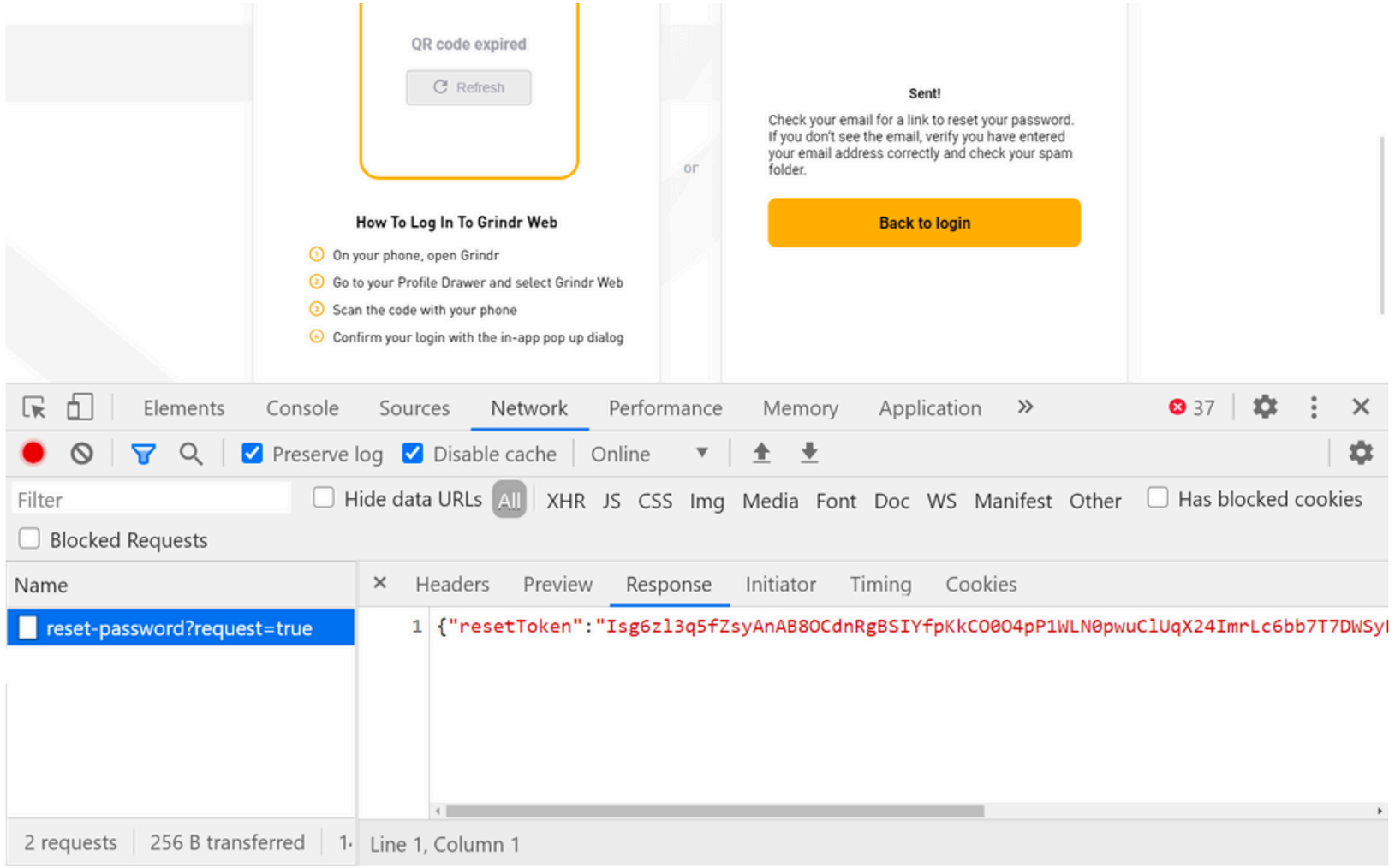
venmo

Grindr (Sept 2020)

https://www.troyhunt.com/hacking-grindr-accounts-with-copy-and-paste/

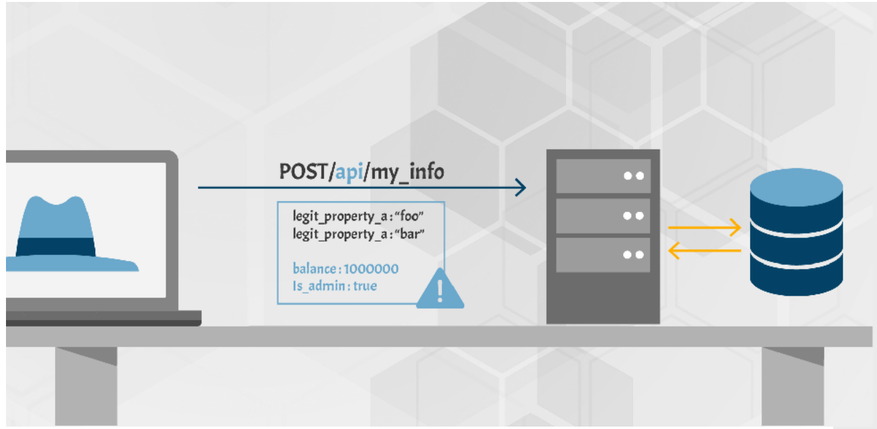
- The Attack
  - Full account takeover for any Grindr account from an email address via password reset
- The Breach
  - Unknown. Company thinks they fixed the issue before anyone could find it.
- Core Issues
  - On password reset, the API leaks the actual reset token which is sent to the user via email (and of course, only the user should know…)

Leaked Data



https://neo-account.grindr.com/v3/user/password/reset?resetToken=Isg6zl3q5fZsyAnAB8OCdnRgBSIYfpKkCO0O4pP1WLN0pwuClUqX24ImrLc6bb7T7DWSyFMG5lREHQmS4CsFR5uh8GEYQxF6Z6V5hsi3vSTuilXzgKRRItwdDIjmSWdq&email=test@scotthelme.co.uk

Mass Assignment (API 6)



Binding client provided data (e.g., JSON) to data models, without proper properties filtering based on an allowlist, usually leads to Mass Assignment. Either guessing objects properties, exploring other API endpoints, reading the documentation, or providing additional object properties in request payloads, allows attackers to modify object properties they are not supposed to.

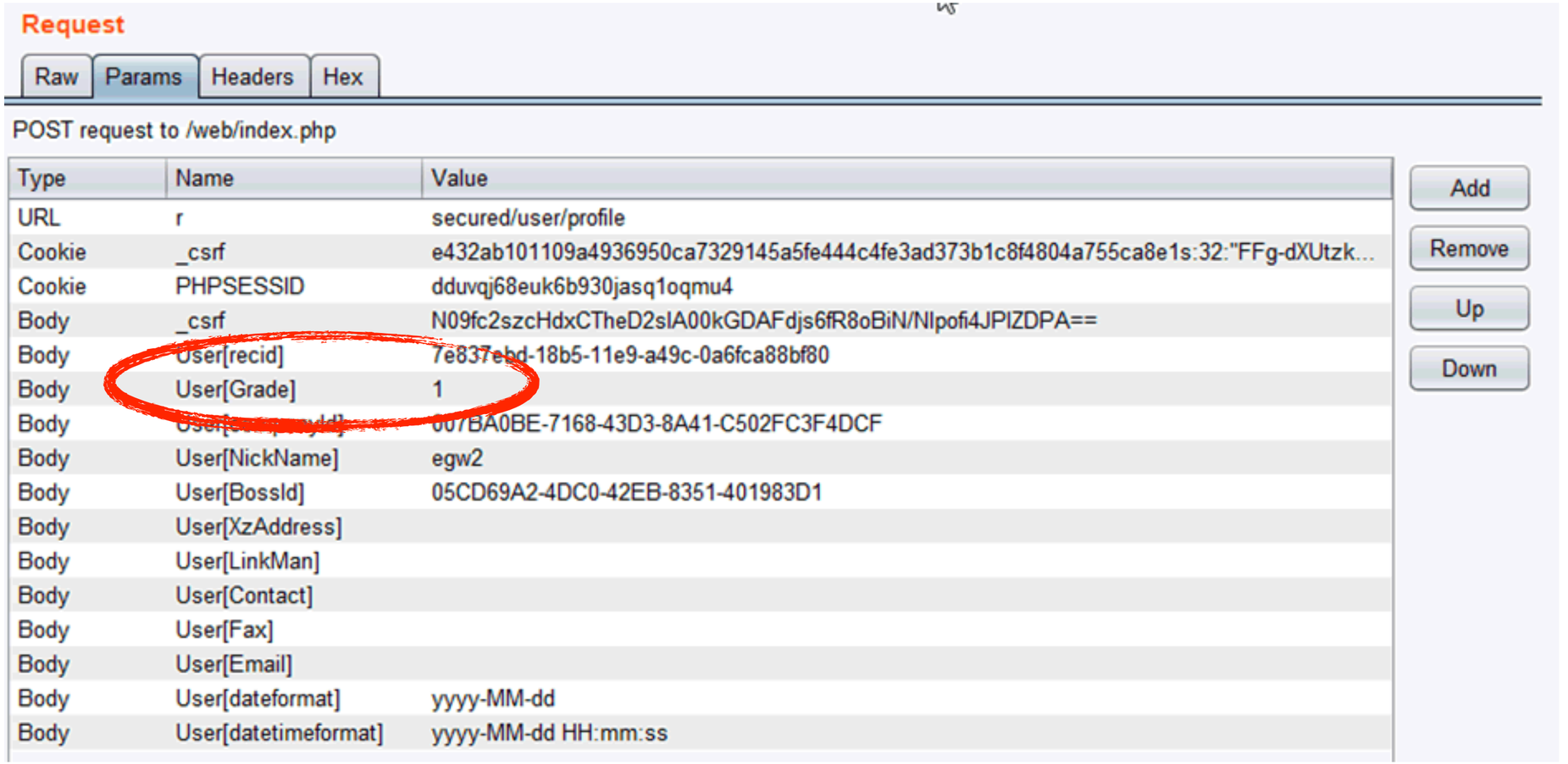
Harbor Registry

GatorWatches (Sept 2019)

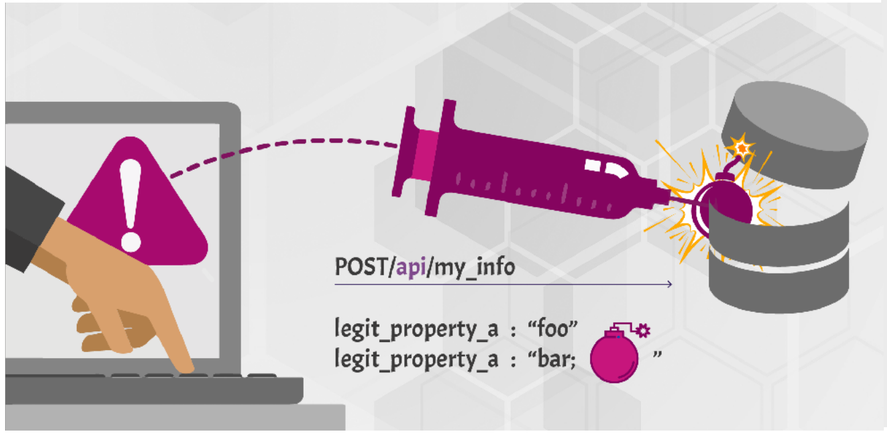
https://www.pentestpartners.com/security-blog/gps-watch-issues-again/

- The Attack
  - Become admin of the Gator platform
- The Breach
  - Admin can see the location of any child wearing the smartwatch
- Core Issues
  - Anyone can set their User[Grade] to `0`, which automatically elevates privileges to admin.

Becoming Admin



Injection (API 8)



Injection flaws, such as SQL, NoSQL, Command Injection, etc., occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

EQUIFAX

Kiwi

EASY2PAY

STARBUCKS (Nov 2019)

https://hackerone.com/reports/592400

- The Attack
  - Blind SQLi leading to RCE (Remote Command Execution)
- The Breach
  - None - This was a bug bounty
- Core Issues
  - SQL Injection allowed to get access to backend production database and execute shell commands on the database server.

And of course…

1. Improper input validation

The primary cause of Log4Shell, formally known as **CVE-2021-44228**, is what NIST calls *improper input validation*.

Loosely speaking, this means that you place too much trust in untrusted data that arrives from outsiders, and open up your software to sneaky tricks based on booby-trapped data.

If you've ever programmed in C, you'll almost certainly have bumped into this sort of problem when using the `printf()` function (*format string and print*).

Normally, you use it something like this:

```
1  int  printf(const char *format, ...);
2
3  int  count;
4  char *name;
5
6  /* print them out somewhat safely */
```

https://nakedsecurity.sophos.com/2021/12/13/log4shell-explained-how-it-works-why-you-need-to-know-and-how-to-fix-it

Input Data Validation

- **No Trust (even for internal APIs and for East-West traffic)**
- Validation can happen client side, but it must happen server-side!
- Do not blindly update data from input structure
  - Apply caution when using frameworks that map directly database records to JSON objects
- **Do not use the same data structures for GET and POST/PUT**
- Validate Inputs
  - Only accept information specified in JSON schema (contract-based, whitelist approach) - Reject all others.
  - Also validate Headers
- **How to test**
  - Send bad verbs, bad data, bad formats, out of bounds, etc.

Output Data Validation

- Never rely on client apps to filter data : instead, create various APIs depending on consumer, with just the data they need
- **Take control of your JSON schemas !**
  - Describe the data thoroughly and **enforce the format at runtime**

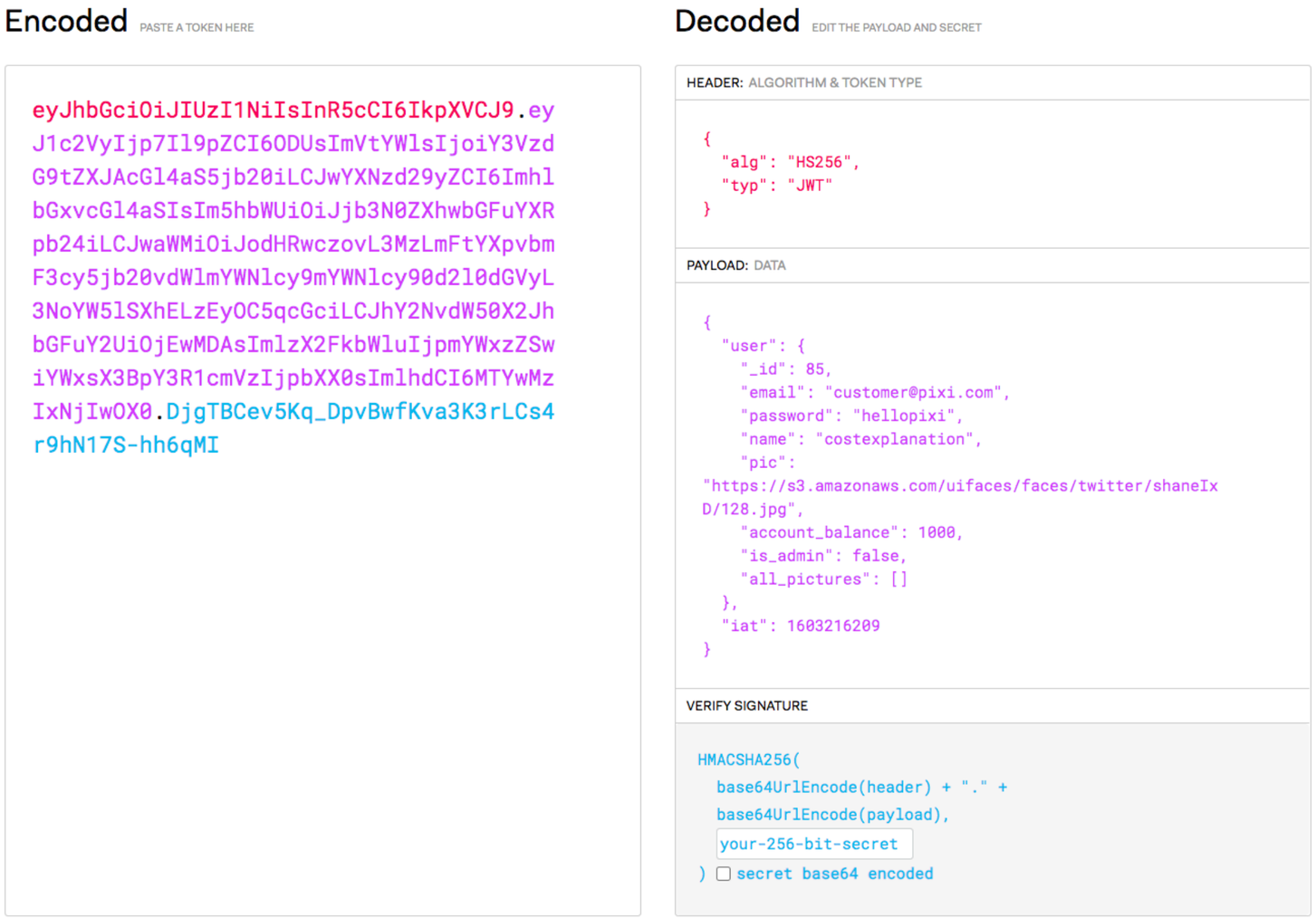
- Review and approve data returned by APIs
- Never expose tokens/sensitive/exploitable data in API responses
- Properly design error messages - Make sure they are not too verbose!
- Beware of GraphQL queries!
  - Validate fields accessed via query

JWTs transport data too!

- Can leak data
- Can be prone to injections (example: kid sql injection)

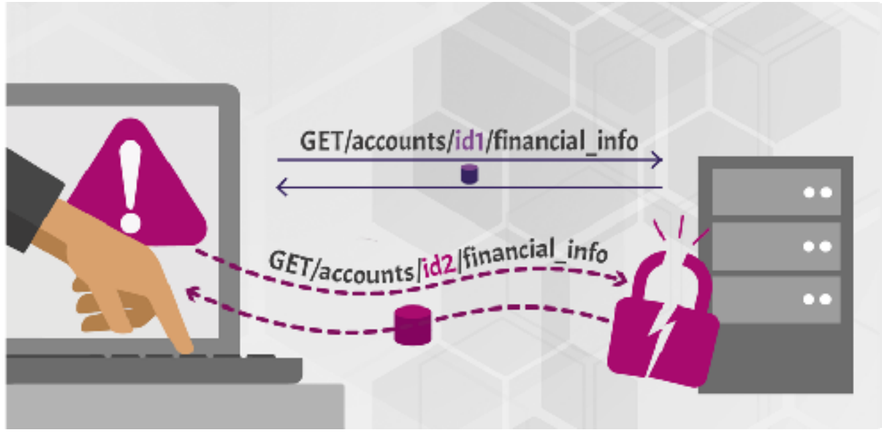
Recommended best practice:

* Use opaque tokens for external consumption
* Use JWTs for internal consumption




AUTHORIZATION

Broken Object Level Access Control (API 1)



APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface Level Access Control issue. Object level authorization checks should be considered in every function that accesses a data source using an input from the user.

Uber

New Relic

Google Cloud

Online Registration System
Ministry of Electronics & Information Technology
Goverment of India

SONICWALL

UBER (SEPT 2019)

https://appsecure.security/blog/how-i-could-have-hacked-your-uber-account

- The Attack
  - Account takeover for any Uber account from a phone number
- The Breach
  - None. This was a bug bounty.
- Core Issues
  - Data leakage 1: driver internal UUID exposed through error message!

```
{
    “status”:”failure”,
    ”data”: {
        “code”:1009,
        ”message”:”Driver ‘47d063f8-0xx5e-xxxxx-b01a-xxxx’ not found”
        }
}
```

  - **Hacker can access any driver, user, partner profile if they know the UUID**
  - Data Leakage 2: Full account information is returned, when only a few fields are used by the UI. This includes the **mobile token** used to login onto the account

Addressing BOLA Issues

- Fine-grained authorisation in **every controller layer**
- Additionally:
  - Avoid guessable IDs (123, 124, 125…)
  - Avoid exposing internal IDs via the API
  - Alternative: GET https://myapis.com/phone/me
- OAuth scopes are not the solution here, as they limit access to an operation and not to a resource.
- Mitigate potential data scrapping by putting rate limiting in place
- **Test this use case (see testing framework from** Yelp **)**

- Great reference: https://inonst.medium.com/a-deep-dive-on-the-most-critical-api-vulnerability-bola-1342224ec3f2


AUTHENTICATION

Broken Authentication (API 2)

Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently. Compromising system's ability to identify the client/user, compromises API security overall.

facebook

Auth0

SIEMENS

Auth0 (April 2020)

https://insomniasec.com/blog/auth0-jwt-validation-bypass

- The Attack
  - Authentication Bypass
- The Breach
  - None. Discovered as part of pen-testing.
- Core Issues
  - The Authentication API prevented the use of **alg: none** with a case sensitive filter. This means that simply capitalising any letter e.g. alg: **nonE**, allowed tokens to be forged.

Addressing Broken Authentication

- No un-authenticated endpoints!
- Define authentication based on **risk**.
- Use short-lived access tokens and limit their scope

- Use correct OAuth grant_types (most likely authorization_code with PKCE)
  - Use the Financial Grade security profiles as reference (https://openid.net/wg/fapi/)
- Make sure you validate JWTs according to Best Practices (RFC 8725) - https://www.rfc-editor.org/rfc/rfc8725.txt

- Use secure storage for credentials
- Watch for tokens in code repos (for example Github Secret Scanning).

- **Test authentication resilience with all kind of combinations!**


RATE LIMITING

Rate/Resources Limiting (API 4)

Quite often, APIs do not impose any restrictions on the size or number of resources that can be requested by the client/user. Not only can this impact the API server performance, leading to Denial of Service (DoS), but also leaves the door open to authentication flaws such as brute force.

Facebook (Feb 2018)

https://appsecure.security/blog/we-figured-out-a-way-to-hack-any-of-facebook-s-2-billion-accounts-and-they-paid-us-a-15-000-bounty-for-it

- The Attack
  - Account takeover via password reset at https://www.facebook.com/login/identify?ctx=recover&lwv=110.
  - facebook.com has rate limiting, beta.facebook.com does not!
- The Breach
  - None. This was a bug bounty.
- Core Issues
  - Rate limiting missing on beta APIs, which allows brute force guessing on password reset code
  - Misconfigured security on beta endpoints

Rate Limiting Recommendations

- Protect all authentication endpoints from abuse (login, password reset, OAuth endpoints)
  - Smart rate limiting : by API Key/access token/user identity/fingerprint
  - Short timespan
- Bad example: Instagram, 200 attempts/min/IP for password reset
  - The list of all potential 6 digits combinations take seconds to generate….

“In a real attack scenario, the attacker needs 5000 IPs to hack an account. It sounds big but that’s actually easy if you use a cloud service provider like Amazon or Google. It would cost around 150 dollars to perform the complete attack of one million codes”

No Authentication
+ No Rate Limiting :
Lethal Combination


LOGGING

Secure Logging

- **Goals**:
  - Forensics
  - Non-repudiation
- **Keep event logs** for anything unusual
  - Rejected requests (auth issues, authorization issues, data validation, application errors).
  - Critical information needs to be logged at the lowest logging level (i.e. not the debug level)
- **Need to record**: what happened, when, who was the caller, where (app/api details, machine name, pod name, etc.)
- **Recommendations**:
  - Log early - Adding logs once code is written is a nightmare...
  - Invest in a shared framework / custom library that everyone uses and which implements those best practices - will make logging easier and coherent

Secure Logging

- Careful with the data we log:
  - No PII
  - No tokens/API Keys
  - No Encryption keys
  - Anything sensitive for your business
- Sensitive data can be :
  - **Masked** (same format/different data)
  - **Hashed** : very useful for tokens/IPs for traceability
  - **Encrypted**
- Logs file may need to be signed for non-repudiation purposes

More at : https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

Tokens/Keys passed as query param!

- Anything in query param end logs in a log somewhere
- …Then pushed to a central log manager…
- …Now it’s visible in dashboards!
- Always use headers to pass sensitive information (or body in a POST)

More at: https://www.fullcontact.com/blog/2016/04/29/never-put-secrets-urls-query-parameters/

CALL TO ACTION!

Use API Top 10 as framework for design and testing

Start worrying about API Security at design time

- ✓ A vulnerability discovered at production time costs up to 30x more to solve

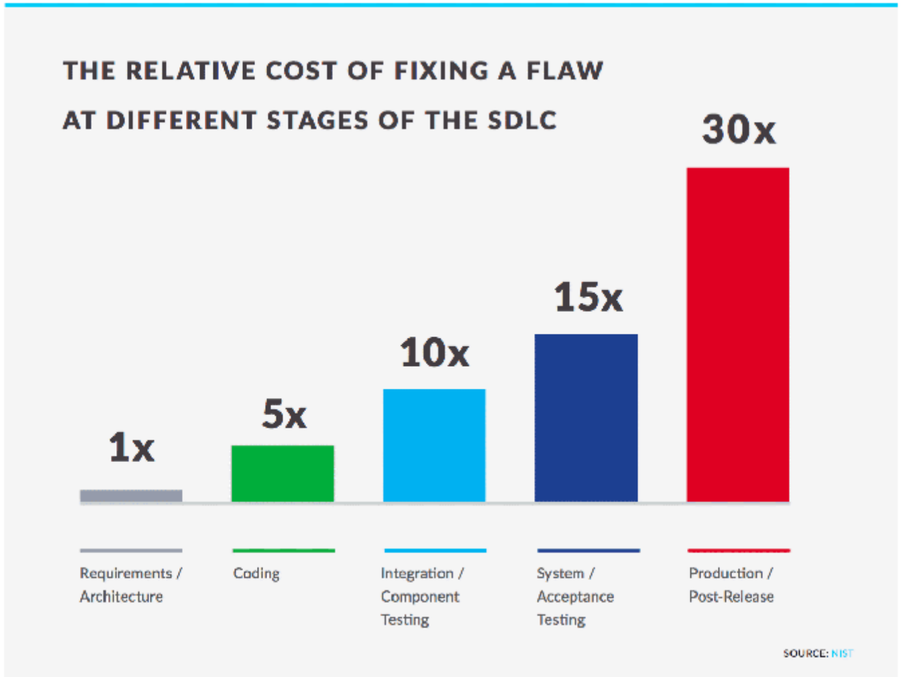
Hack yourselves leveraging API contracts

- ✓ 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

References

- apisecurity.io
- **Input Validation**
  - https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
- **Defense in-depth**
  - http://searchsecurity.techtarget.com/definition/defense-in-depth
- **OWASP REST Security Cheat Sheet**
  - https://www.owasp.org/index.php/REST_Security_Cheat_Sheet
- **Transport Layer Security Cheat Sheet**
  - https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet
  - https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- **HTML5 Security Cheat Sheet**
  - https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet#Local_Storage