API Security
API Owner

Escuela de APIs

{api addicts}

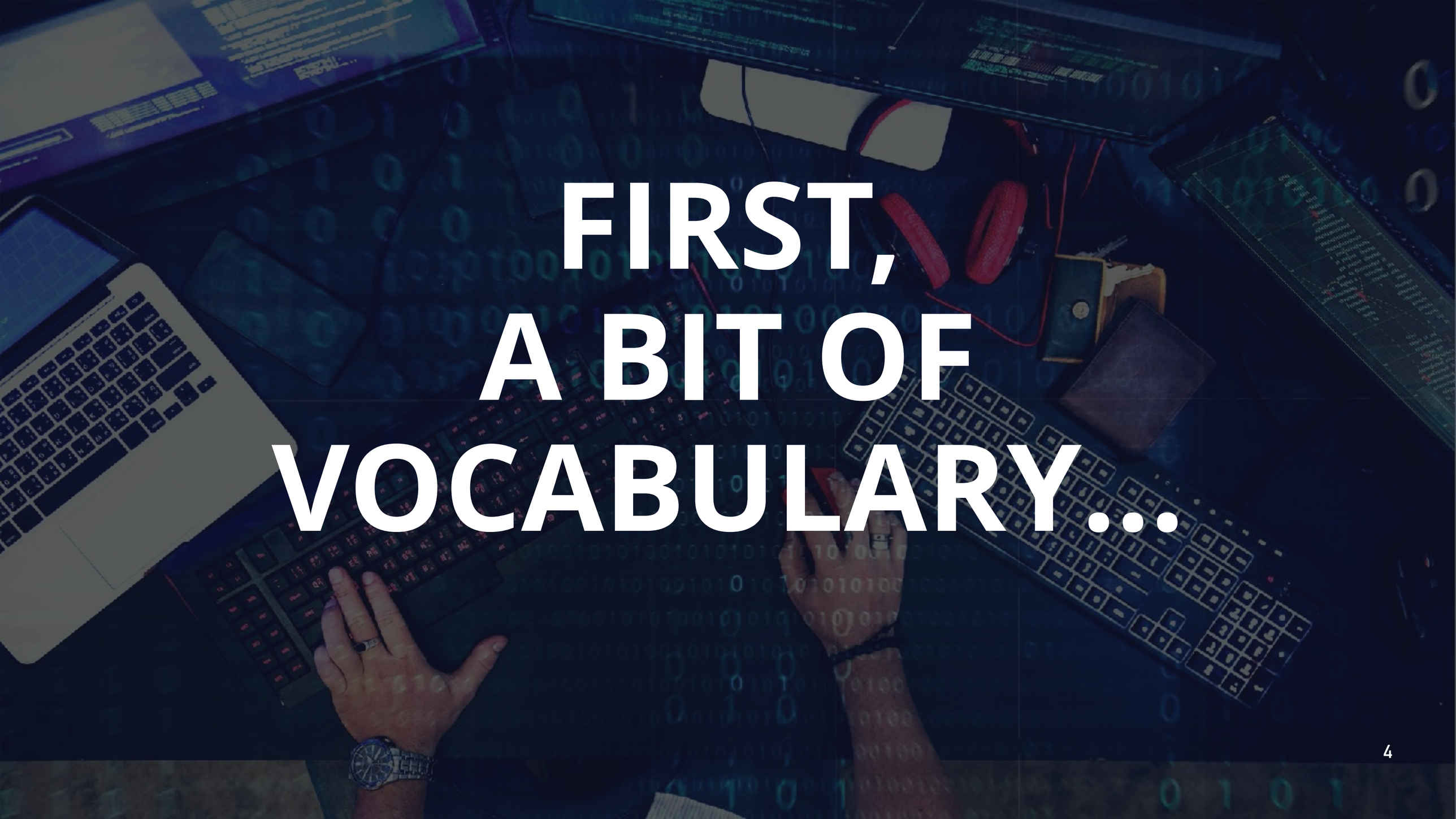
Isabelle MAUNY

- ➔*Field CTO and co-founder @42Crunch*
- ➔*42Crunch delivers an API Security Platform*
- ➔*We are the company behind apisecurity.io*
- ➔*Working with APIs and integration since 2005!*
- ➔*French native, in Spain since 2003*

{api addicts}

Agenda

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



FIRST, A BIT OF VOCABULARY…

Security has two different aspects

- Putting in place a security architecture achieving **security goals**
- Preventing from **attacks** against this security architecture.

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.

Attacks and countermeasures

ASSET	A resource of value such as the data in a database or on the file system, or a system resource (such as an API)
VULNERABILITY	A weakness or gap in security program that can be exploited by threats to gain unauthorised access to an asset.
THREAT	Anything that can exploit a vulnerability and obtain, damage, or destroy an asset.
ATTACK/EXPLOIT	A threat in action, to harm an asset.
COUNTERMEASURE	A safeguard that addresses a threat and blocks attacks.

Security is a Layered Approach

APPLICATION SECURITY	Measures taken throughout the code's life-cycle to prevent gaps in the security policy of an application, i.e. flaws in the design, development, execution of an application
SYSTEM SECURITY	Control of access to a computer system's resources, specially its data and operating system files.
NETWORK SECURITY	Policies and practices adopted to prevent and monitor un-authorized access, misuse, modification, or denial of a computer network and network-accessible resources.

API Breaches are on the rise!

- 350+ 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

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

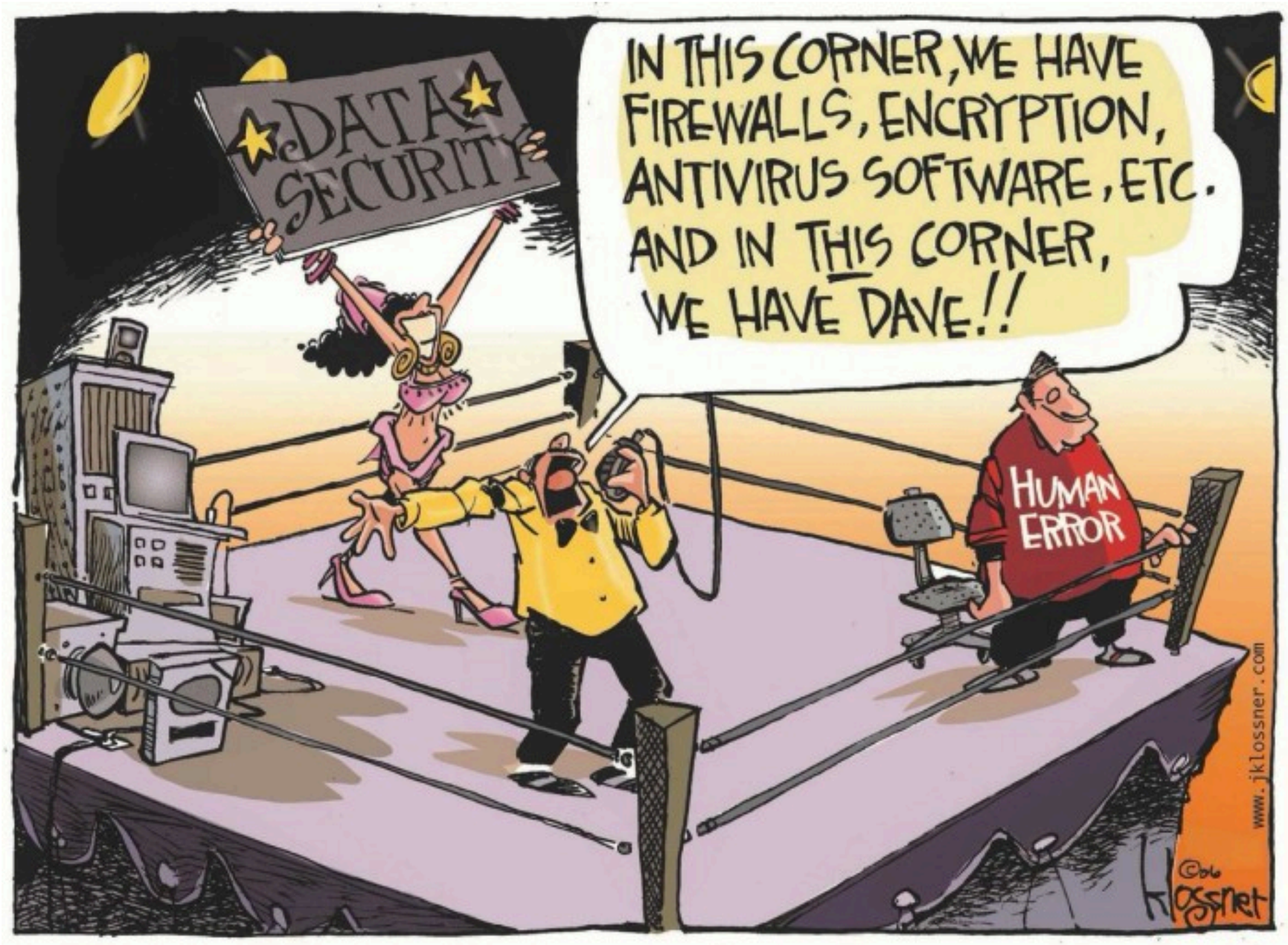
Equifax - 147 million users personal data stolen

Starbucks - 100 million customer records accessed

Verizon - 14 million subscribers phone numbers and PINs exposed

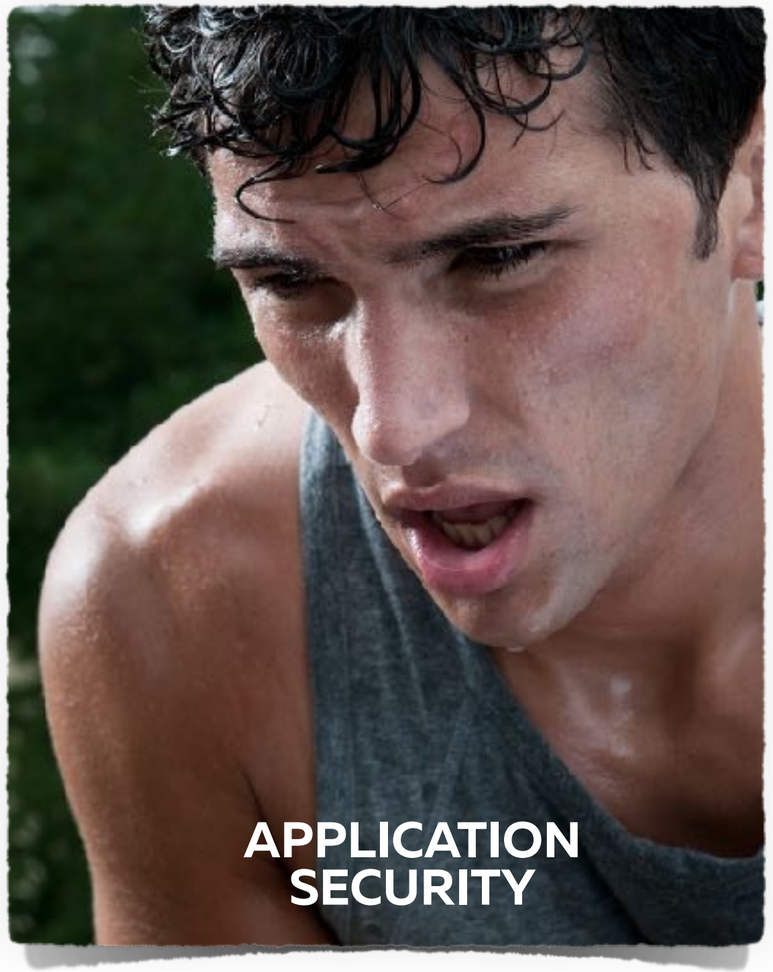

WHY IS THIS
HAPPENING?

WE ARE HUMANS

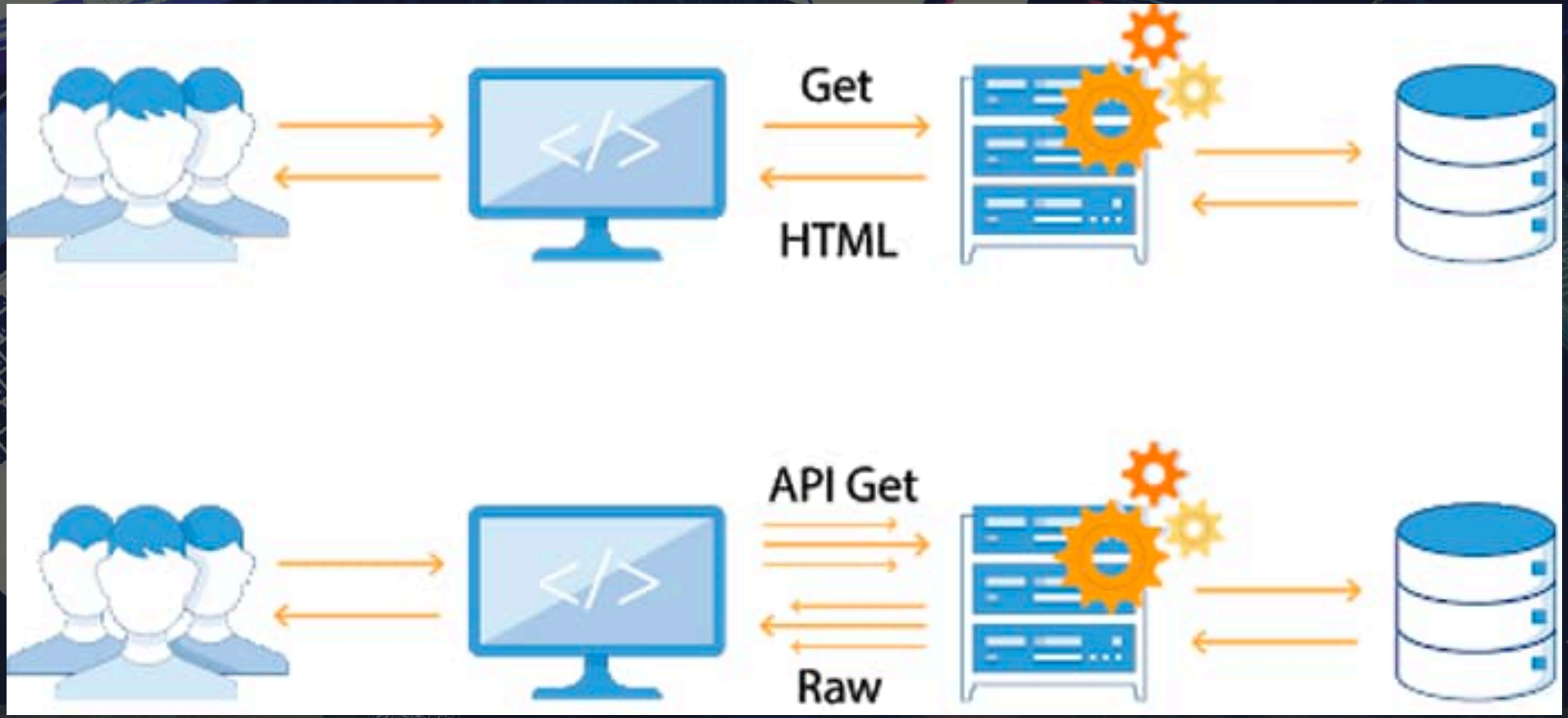


Security is considered too late!

MANY APIS, MANY DEPLOYMENTS



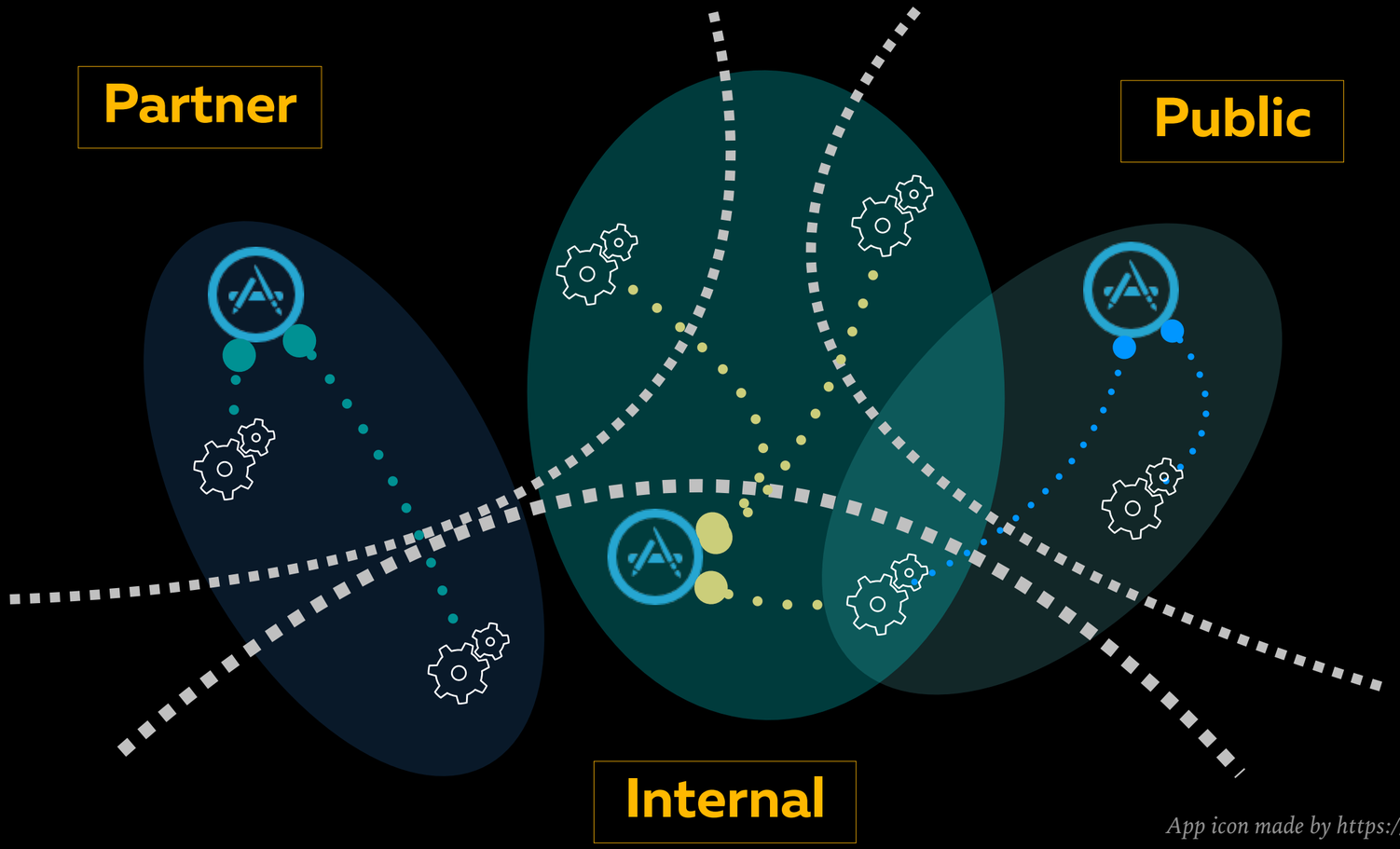
Applications Architecture has changed!



Source: https://apisecurity.io/encyclopedia/content/owasp/owasp-api-security-top-10.htm

“The perimeter has disappeared. **It’s no longer about protecting boundaries, it’s about protecting data.**”

Virtual application networks make security even harder



App icon made by https://www.flaticon.com/authors/pixel-buddha

OWASP API Security Top 10

APIs have different vulnerabilities

- 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

SECURITY PRINCIPLES

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

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

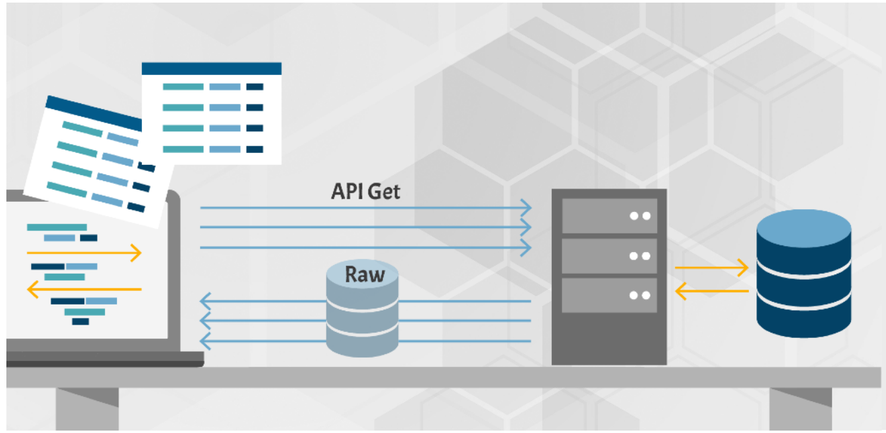
All APIs need to be treated as public

API Security is risk based

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


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

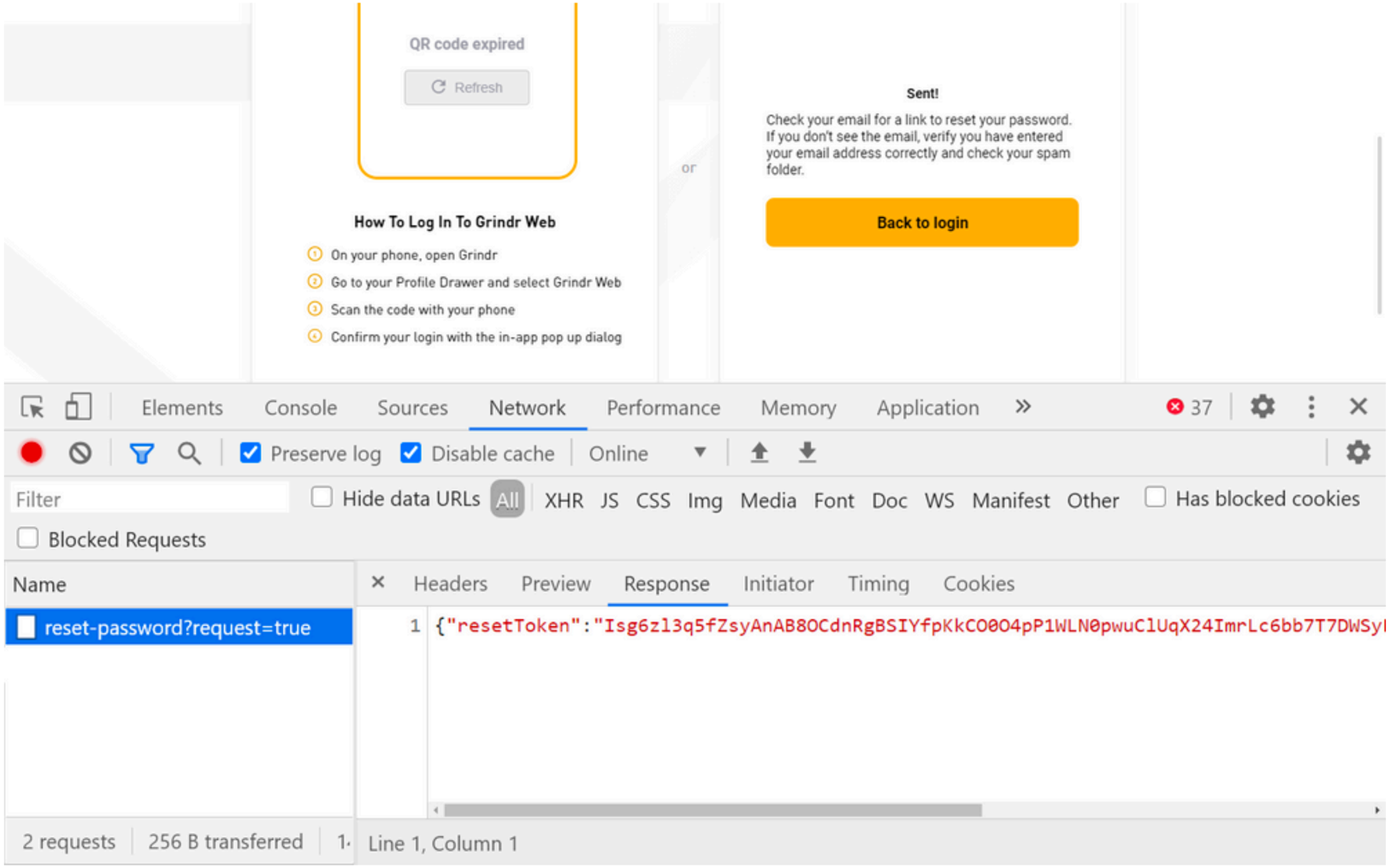
3Fun הליכוד Likud Party jCrush venmo

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

Another API3 vector: JWTs!

- Recommended best practice:
  - * Use opaque tokens for external consumption
  - * Use JWTs for internal consumption

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J1c2VyIjp7Il9pZCI6ODUsImVtYWlsIjoiY3Vzd
G9tZXJAcGl4aS5jb20iLCJwYXNzd29yZCI6Imhl
bGxvcGl4aSIsIm5hbWUiOiJjb3N0ZXhwbGFuYXR
pb24iLCJwaWMiOiJodHRwczovL3MzLmFtYXpvbm
F3cy5jb20vdWlmYWNlcy9mYWNlcy90d2l0dGVyL
3NoYW5lSXhELzEyOC5qcGciLCJhY2NvdW50X2Jh
bGFuY2UiOjEwMDAsImlzX2FkbWluIjpmYWxzZSw
iYWxsX3BpY3R1cmVzIjpbXX0sImlhdCI6MTYwMz
IxNjIwOX0.DjgTBCev5Kq_DpvBwfKva3K3rLCs4
r9hN17S-hh6qMI
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

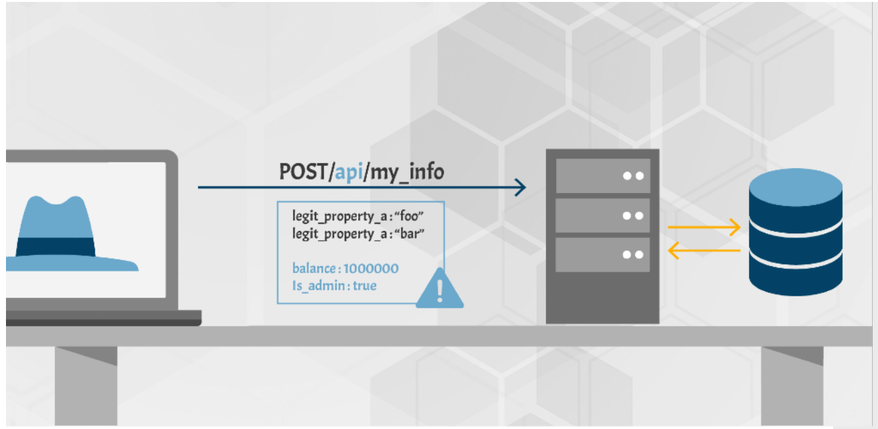
PAYLOAD: DATA

```
{
  "user": {
    "_id": 85,
    "email": "customer@pixi.com",
    "password": "hellopixi",
    "name": "costexplanation",
    "pic":
"https://s3.amazonaws.com/uifaces/faces/twitter/shaneIx
D/128.jpg",
    "account_balance": 1000,
    "is_admin": false,
    "all_pictures": []
  },
  "iat": 1603216209
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

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

myGator -The start of life's adventures

Uber

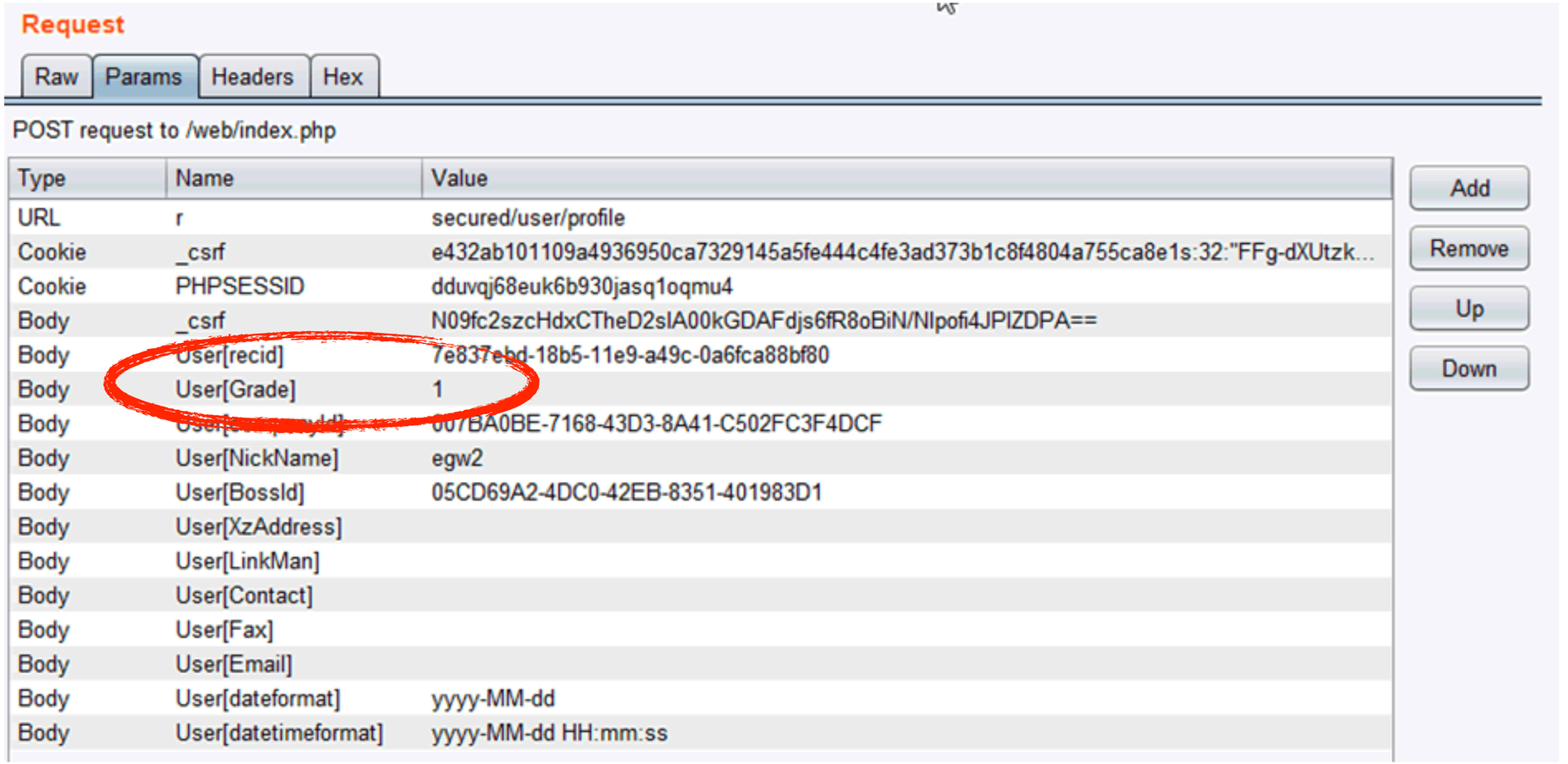
New Relic

GatorWatches (Sept 2019)

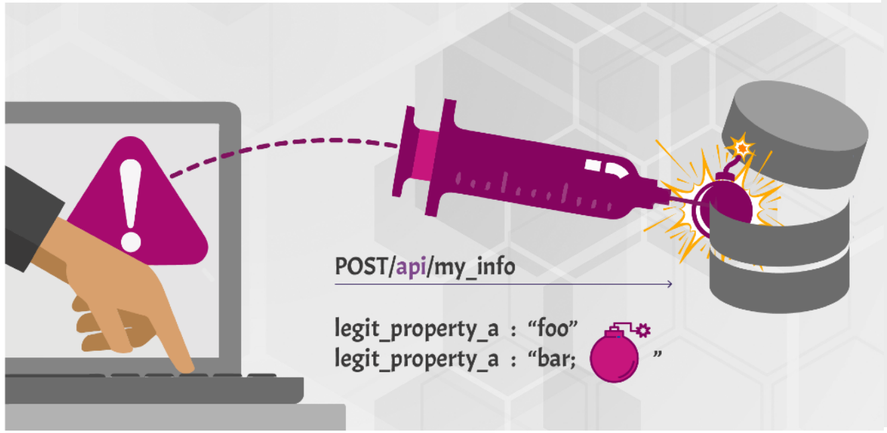
https://www.pentestpartners.com/security-blog/gps-watch-issues-again/

- 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

{api addicts}

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.

Input Data Validation

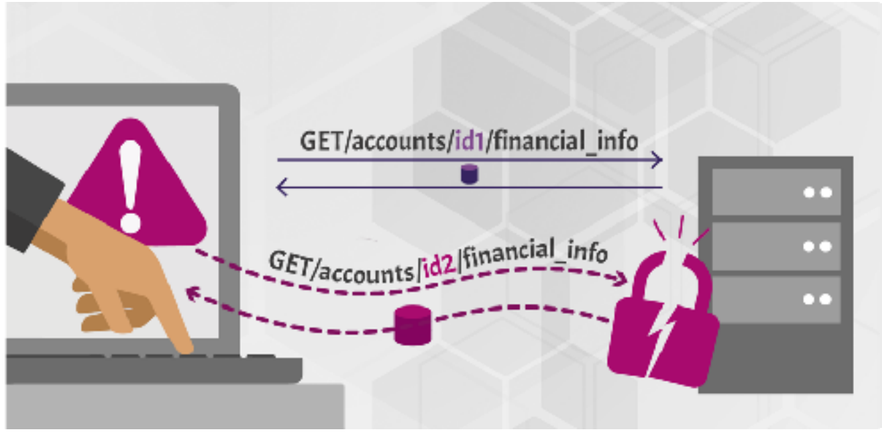
- **No Trust (even for internal APIs and for East-West traffic)**
- Validation can happen client side, but it must happen server-side!
- Do not blindly update data from input structure
  - Apply caution when using frameworks that map directly database records to JSON objects
- Do not use the same data 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

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
- Beware of GraphQL queries!
  - Validate fields accessed via query


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.
- **Test this use case (see IDOR testing from** Yelp **)**
- Mitigate potential data scrapping by putting rate limiting in place


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!
- Prefer OpenID Connect to Basic Auth (API Keys, like OAuth token do **not** allow to authenticate the user)
- Use short-lived access tokens and limit their scope
- Use OAuth properly (most likely authorization_code with PKCE)
  - Financial API Grade 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

- On App Side :
  - Enforce 2FA, captcha
  - Use secure storage for credentials

- **Test 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

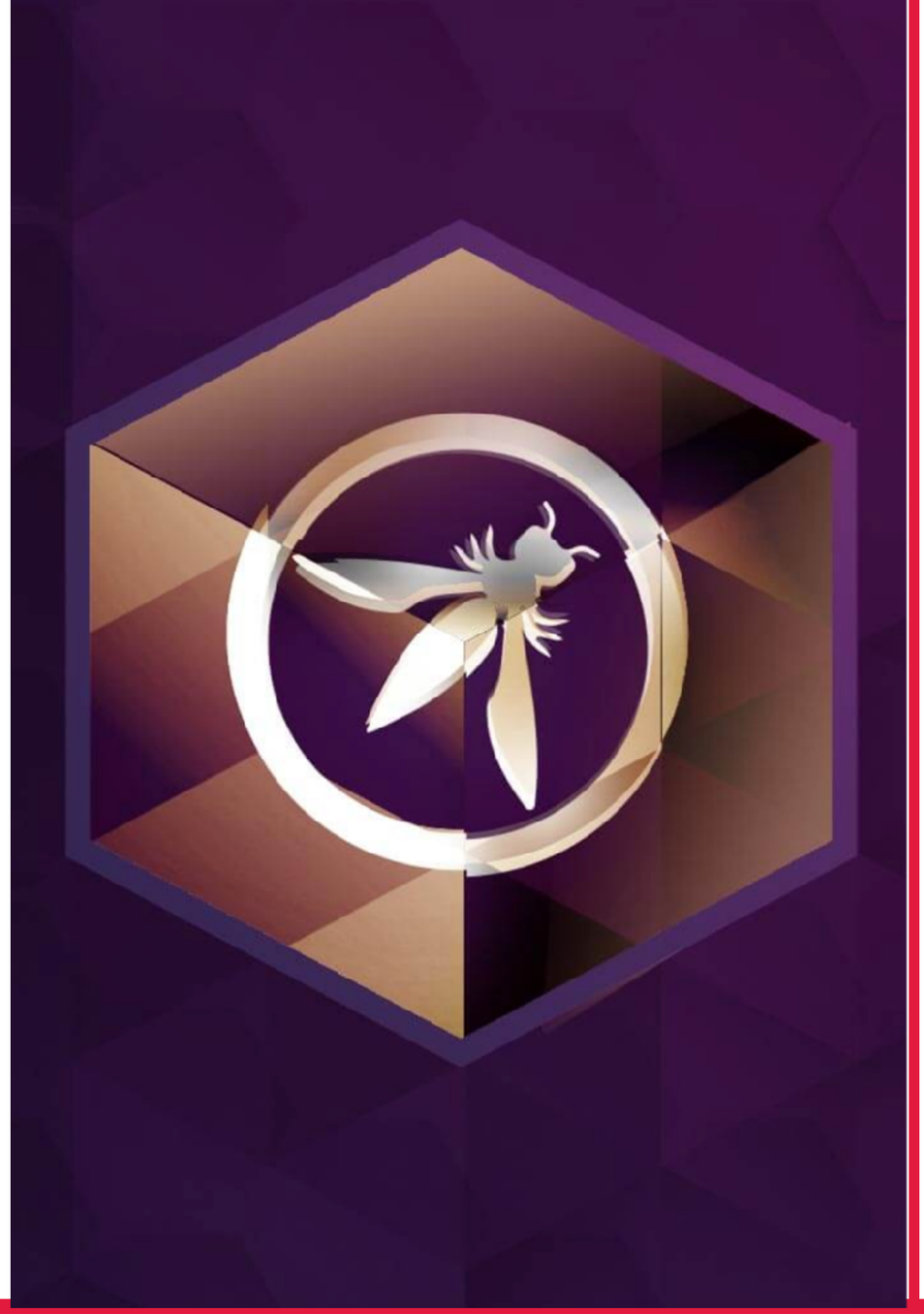
- The Attack
  - Account takeover via password reset at https://www.facebook.com/login/identify?ctx=recover&lwv=110.
  - facebook.com has rate limiting, beta.facebook.com does not!
- The Breach
  - None. This was a bug bounty.
- Core Issues
  - Rate limiting missing on beta APIs, which allows brute force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

Rate Limiting Recommendations

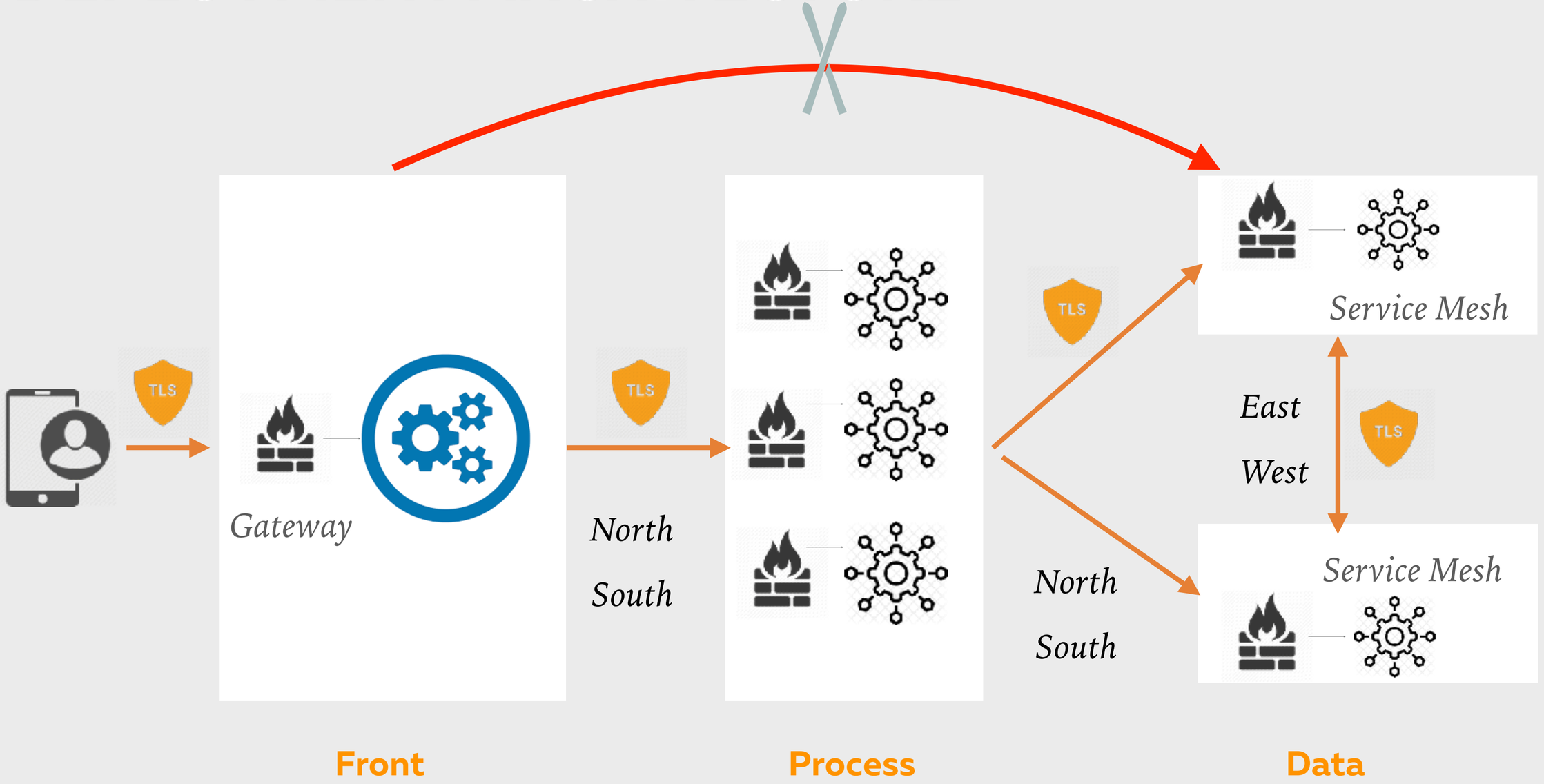
- Protect all authentication endpoints from abuse (login, password reset, OAuth endpoints)
  - Smart rate limiting : by API Key/access token/user identity/fingerprint
  - Short timespan
- Bad example: Instagram, 200 attempts/min/IP for password reset
  - The list of all potential 6 digits combinations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

API 2 + API 4 :
Lethal Combination



DEPLOYMENT ARCHITECTURE



App icon made by https://www.flaticon.com/authors/pixel-buddha

Final Recommendations

- 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!
  - 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.

References

- **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

API Security Deep Dive Preview

- OWASP Top 10 Deep Dive with examples
- OAuth and OpenID Connect: Architecture and Demos
- JWT Primer / JWT Best Practices
- Hacking Demo
- Secure APIs development
  - Sample JS Application
  - OpenAPI Usage
- Tooling Landscape
  - API Management
  - WAFs
  - Libraries Analysis (SCA)
  - Container Security

DATOS DE CONTACTO

+34 **91 764 79 82**

contacta@apiaddicts.org

www.apiaddicts.org

Facebook

ApiAddicts

Linkedin

API Addicts

Twitter

@APIAddicts

Meet Up

API Addicts

Youtube

API Addicts

{api addicts}