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}

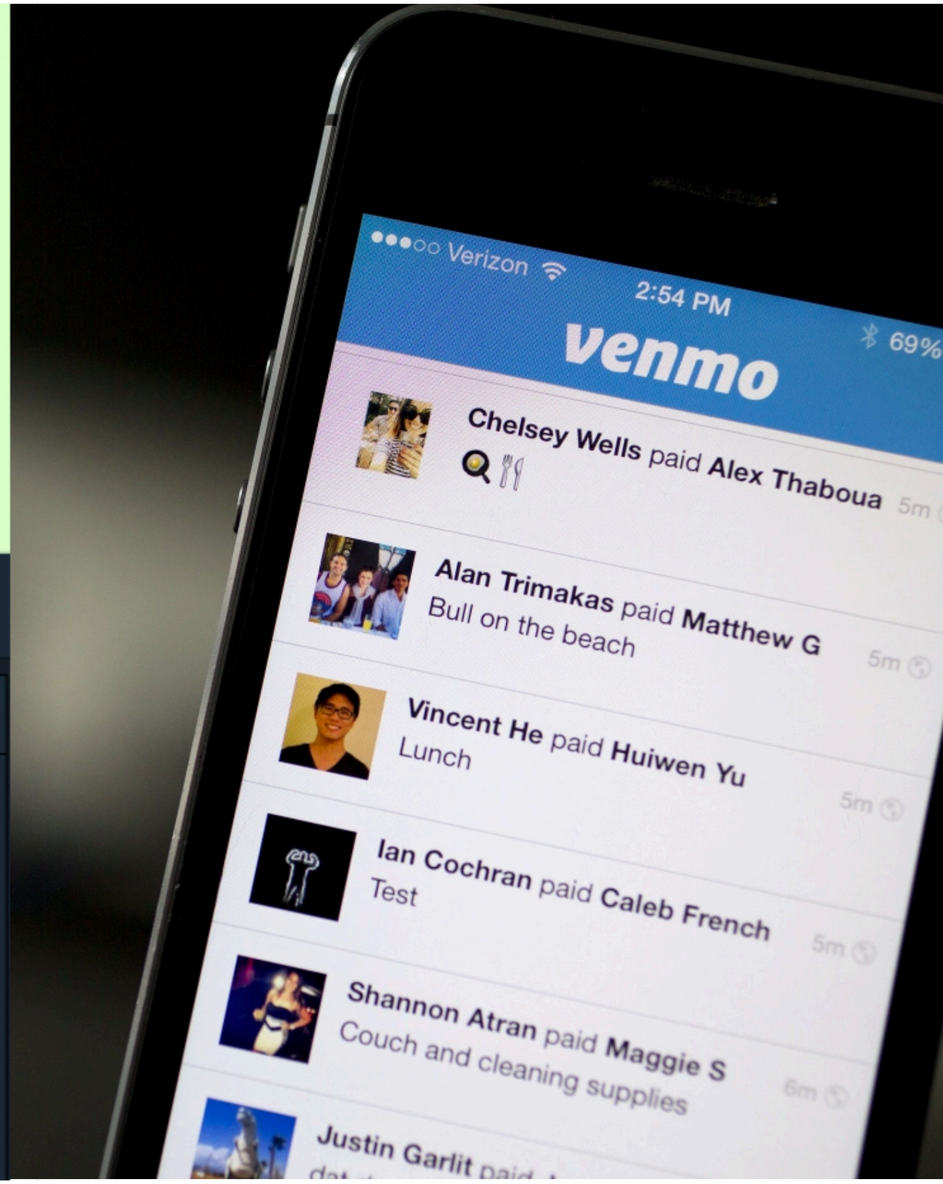
APIs connect the world

Data is the new gold!

APIs are a critical path to data

Equifax
Experian
Verizon
T-Mobile
Facebook
LinkedIN
Parler

Guys in hoodies are interested!



Open Banking pushed a bit too far…

Healthcare

Playing with FIHR (HL7)

55 Healthcare Apps
All vulnerable.

Casino hacked via fish tank

Secure your laptop. Secure your smart phone. Secure your tablet. And, before I forget, secure your fish tank. Yes, you heard me. Your fish tank.

Not a week passes by without an API vulnerability reported on apisecurity.io, and this since Oct. 2018!

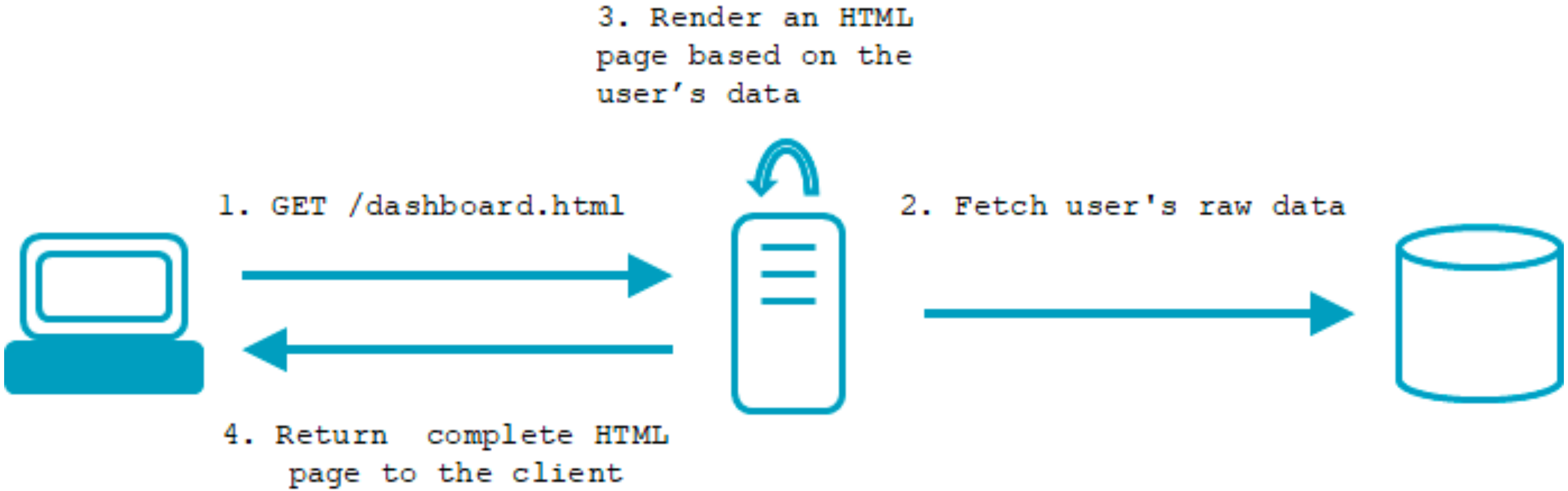
Recurring Combination of:

- Un-authorized data access
- API abuse
- Data leakage
- Authentication failures

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

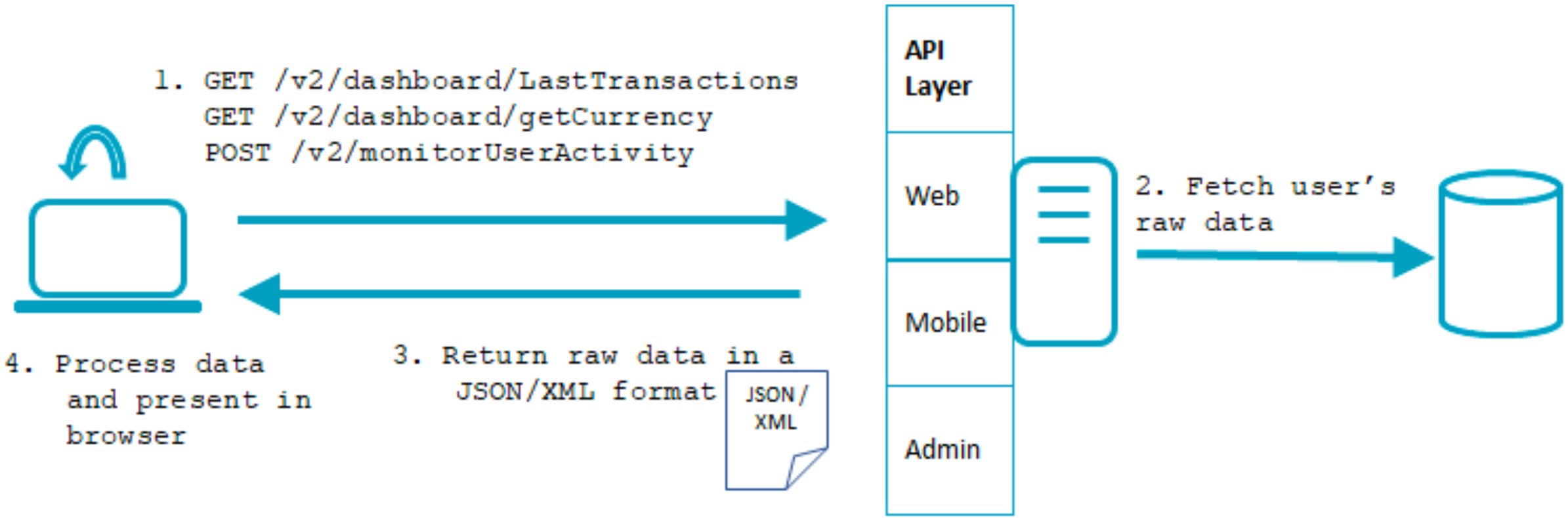
WHY ARE APIS SUCH A PROBLEM?

“Traditional” web architecture



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

API-based API architectures



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

Security Architectures
are built to protect this.

With APIs, security measures need to protect this!

Development plays hard to catch…

APPLICATION DEVELOPMENT

APPLICATION SECURITY

Security is still an afterthought!

Start testing security scenarios **as early as possible**

Design Flaws

APIs suffer from many design flaws, which are hard, including impossible to fix after the fact.

Before doing an API. **Stop and Think.**

Application Security is **hard**!

For everyone. Invest in **education**.

A BIT OF VOCABULARY

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

How ?

- Define which APIs are the most sensitive, for example:
  - You’re likely to be in the news if something bad happens
  - Your reputation will be affected
- **Where** is data stored, **how** it is accessed and by **who** ?
- What are the potential threats and how do we address them?
- STRIDE model
  - Invented in 1999 by Microsoft, but still relevant
  - About knowing the threats and how you mitigate them and where
- Sample : Irius Risk (Demo)

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

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

How ?

- Central Management of resources
- Manage API Lifecycle
  - Create
  - Retire
  - Version
- Protect against zombie APIs!

- Shadow APIs are created by Shadow IT
- Evaluate your API posture
  - Where are my APIs ?
  - Who has access to them ?
  - How are they protected ?

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

DOWNLOAD

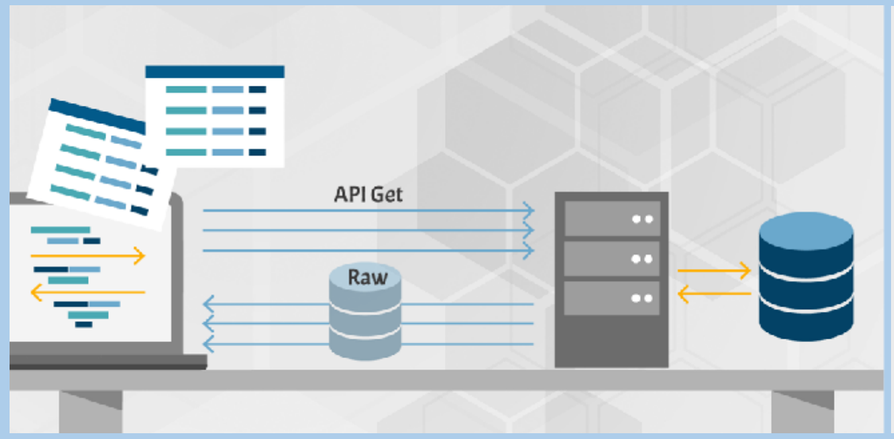
Data Protection Auth / Authorization Governance/Operations

{api addicts}

Testing Pixi

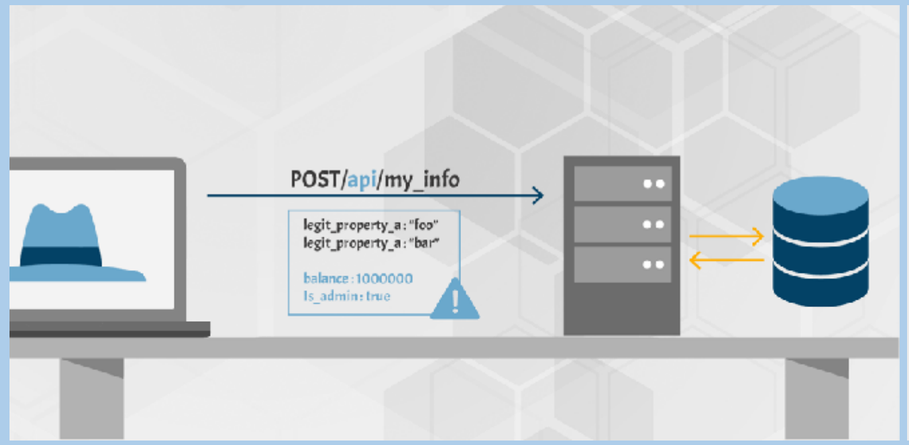
- Shared API at https://178.128.139.33/ (self-signed certs)
- Github project: https://github.com/isamauny/apisecurity-training
  - Local build with docker compose
  - Postman Collection

DATA PROTECTION



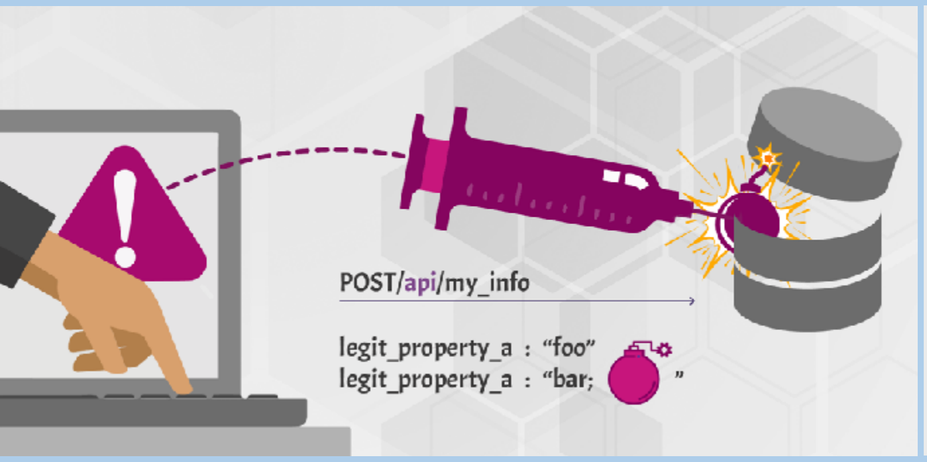
Mass Assignment (API 6)

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.



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.



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.

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.

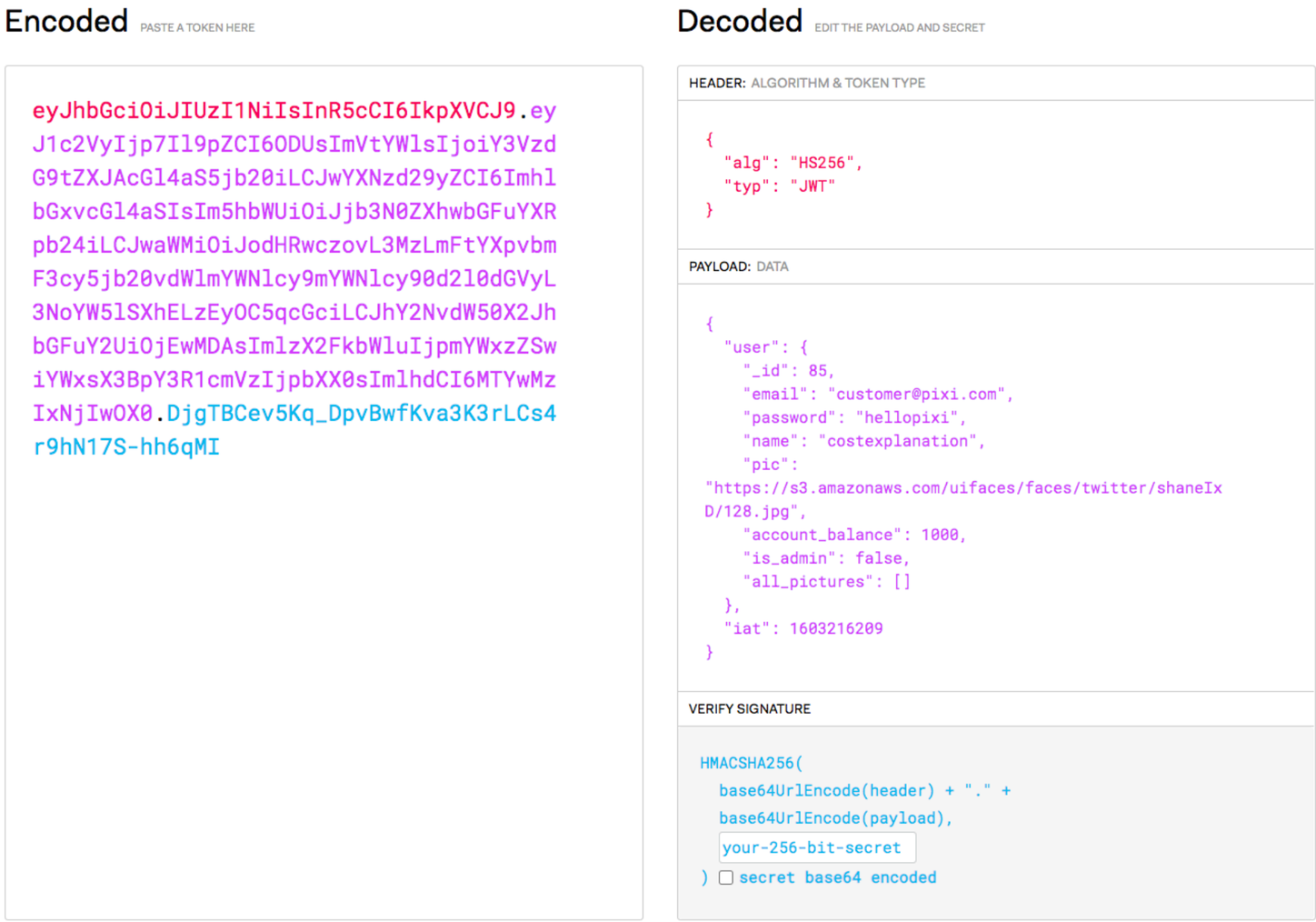
Output Data Validation

- 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

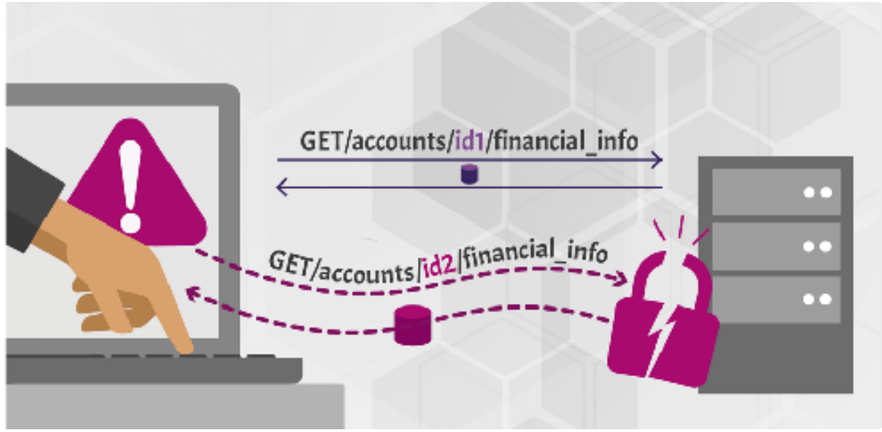
JWTs transport data too!

- 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

COINBASE (FEB 2022)

https://www.coinbase.com/blog/retrospective-recent-coinbase-bug-bounty-award

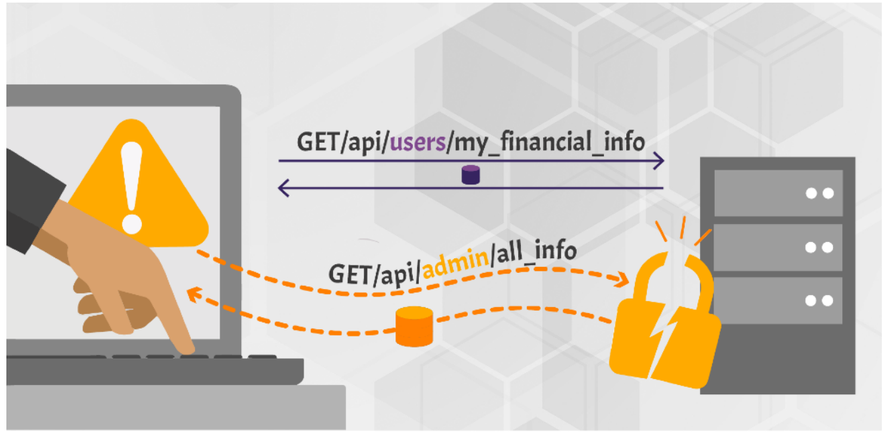
- The Attack
  - Hacker can achieve successful transactions with an empty wallet
- The Breach
  - None. $250.000 bug bounty !!
- Core Issues
  - A user has an account with 100 SHIB, and a second account with 0 BTC.
  - The user submits a market order to the BTC-USD order book to sell 100 BTC, but **manually edits their API request to specify their SHIB account as the source of funds.**
  - Here, the validation service would check to determine whether the source account had a sufficient balance to complete the trade, but not whether the source account matched the proposed asset for submitting the trade.
  - As a result, a market order to sell 100 BTC on the BTC-USD order book would be entered on the Coinbase Exchange.

Addressing BOLA Issues

- **True fix** : Fine-grained authorisation to resources in **every controller layer**
  - Decision engine is required, preferably externally to the code.
  - Need to be able to enforce something like : *Jean is allowed to access account 123, but cannot edit or delete it*.
- **OAuth scopes *are not* the solution here, as they limit access to an operation and not to a resource.**
- Potential mitigation approaches
  - Avoid using IDs when possible: GET https://myapis.com/phone/me
  - Avoid iterative IDs (123, 124, 125…)
  - Avoid exposing internal IDs through your APIs
  - Mitigate potential data scrapping by putting **rate limiting** in place (and alerting!)

- **Test this use case!**
- **Learn more:** https://inonst.medium.com/a-deep-dive-on-the-most-critical-api-vulnerability-bola-1342224ec3f2

API5 : Broken Function Level Authorization



Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws. By exploiting these issues, attackers gain access to other users' resources and/or administrative functions.

הליכוד
Likud Party

Likud Voting App

https://www.zdnet.com/article/netanyahus-party-exposes-data-on-over-6-4-million-israelis/

- The Attack
  - Data breach using leaked admin userid/passwords
- The Breach
  - Unknown. Potentially could have leaked the details of 6.4 million Israelis.
- Core Issues
  - Admin endpoint left open with no authentication
  - Allows to retrieve all systems users, including the password (in clear).
  - **Endpoint was hardcoded in application source code**.

Addressing API 5

- Design properly your authorization policies and test them !
  - OAuth scopes can be used to restrict the access to operations.
- Do not mix admin and non-admin operations in the same API
  - Easy to discover via dictionary attacks
- Restrict access to admin APIs, for example:
  - by Mutual TLS
  - by IP Range
  - Do not rely on the client to do that!

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

Choosing an Authentication Method

- Define a Threat model to decide what is best for your API
- Choose authentication method balancing **risk and ease of use**: who are the consumers ? What is the data sensitivity ?
- **Do not use API keys or bearer tokens for authentication**
  - Remember a Bearer token is like a hotel key, once you have it, you can open the door. There is no way to validate who you are.

Which Options are available ?

Type	Format	Potential Issues	Advantages
Basic Auth	b64 encoded User/password	Goes in clear (encoded) over the network Deprecated.	Backward compatibilty
API Key	Long string of characters	Travels over network Prone to leaks (Github, hardcoded in apps etc.) Expiration management	Simple to use Opaque
OAuth Access Token	Long string of characters JWT	Used for authentication Complexity Carries data	Delegated authorization Reduced security risks Expires
OpenID Connect	JWT	Complexity	True Authentication Session/Logout management Reduced Security Risks
Signatures	Digital signature in header/body	Complex for API consumers	Integrity of information API/Signing keys are not travelling over network.

Addressing Broken Authentication

- Current Open Banking approaches
  - Use OpenID Connect as per the FAPI profiles
    - https://openid.net/specs/openid-financial-api-part-1-1_0-final.html
    - https://openid.net/specs/openid-financial-api-part-2-1_0.html
  - Use multi-factor authentication
- Special attention to authentication and password resets endpoints
  - Specific rate limiting and brute force prevention
  - Same format message for all potential authentication issues (bad user or bad password)

- **Test authentication resilience with all kind of combinations!**
- **Learn more: https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication**

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

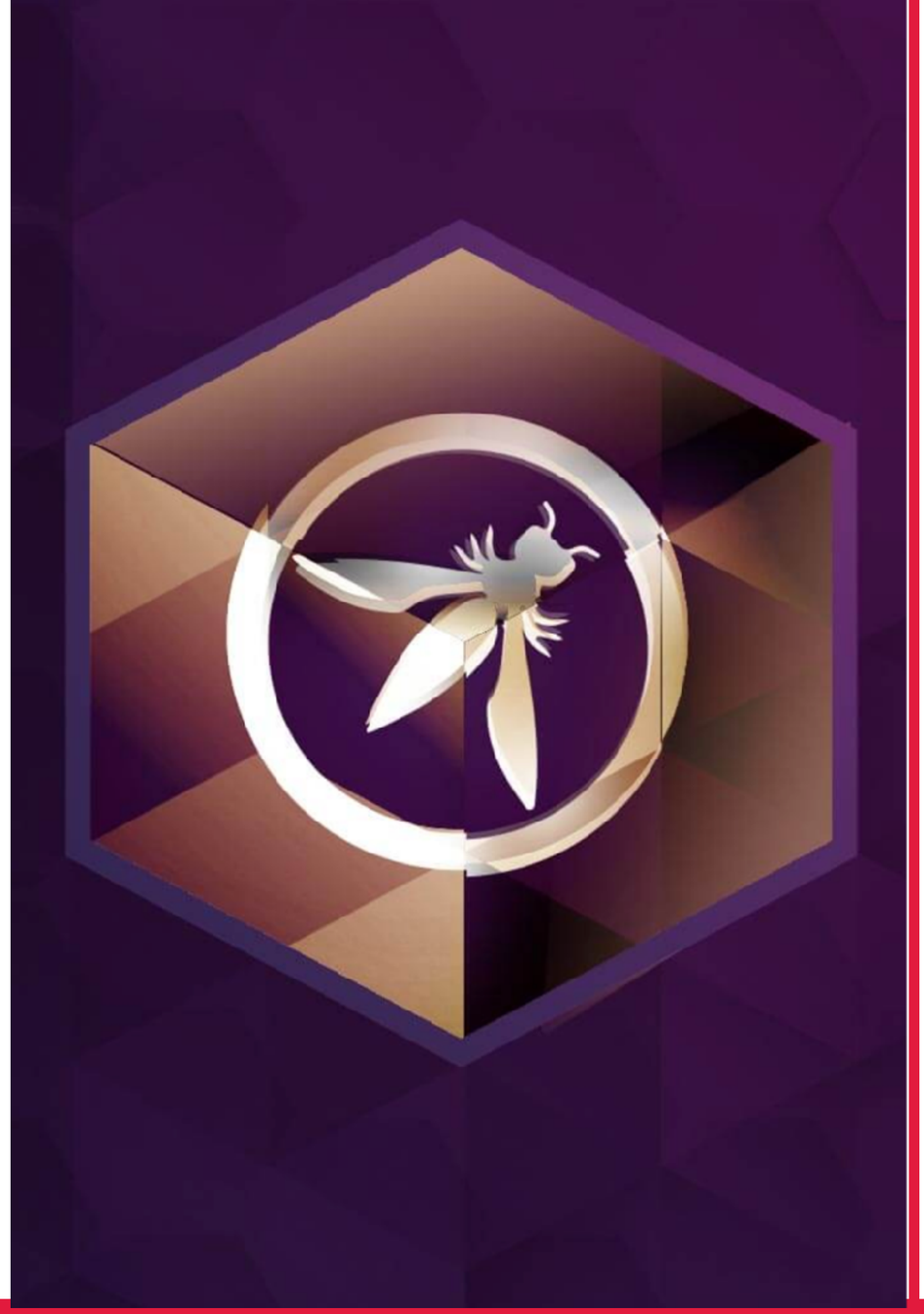
- The Attack
  - Account takeover via password reset at https://www.facebook.com/login/identify?ctx=recover&lwv=110.
  - facebook.com has rate limiting, beta.facebook.com does not!
- The Breach
  - None. This was a bug bounty.
- Core Issues
  - Rate limiting missing on beta APIs, which allows brute force 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
- **Need 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 :
  - **Redacted** (replaced with xxxxx for example)
  - **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/


WE NEED
PROACTIVE
SECURITY

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

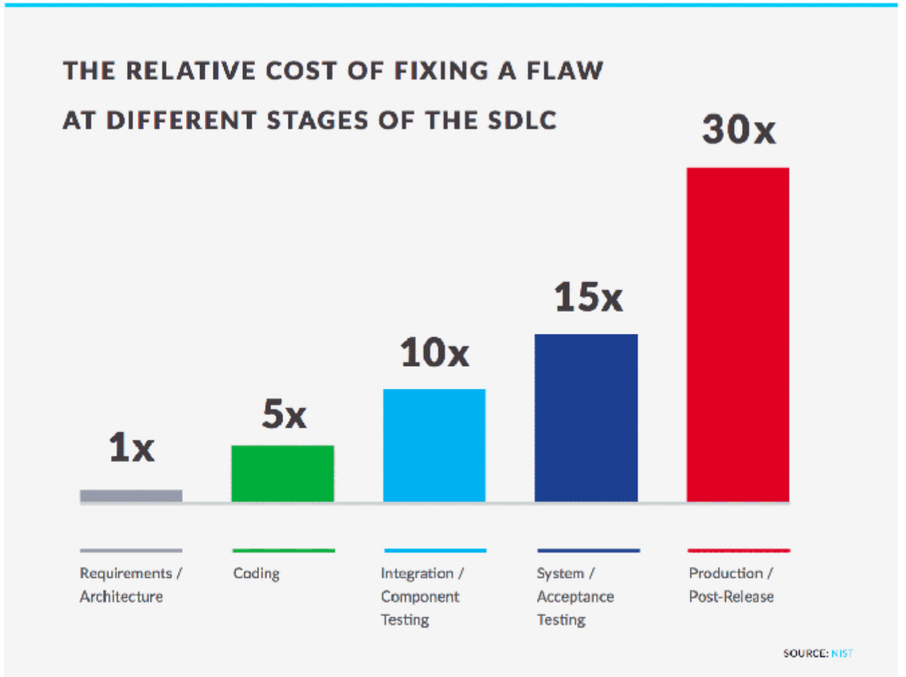
Hack yourselves leveraging API contracts

- ✓ 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/

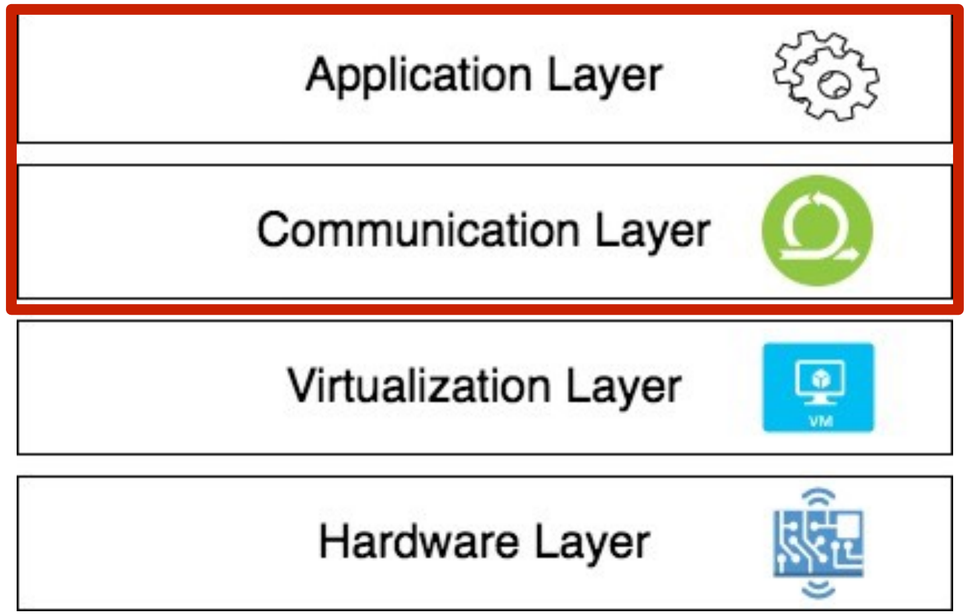


"I think security, in most cases, is not a single person's specialization. Security must be a practice of every member of the team from the frontend developer to the system administrator (also non tech roles)."

From: Gitlab DevSecOps report - 2021

ARCHITECTURE AND TOOLS

API Security Applies at Multiple Levels



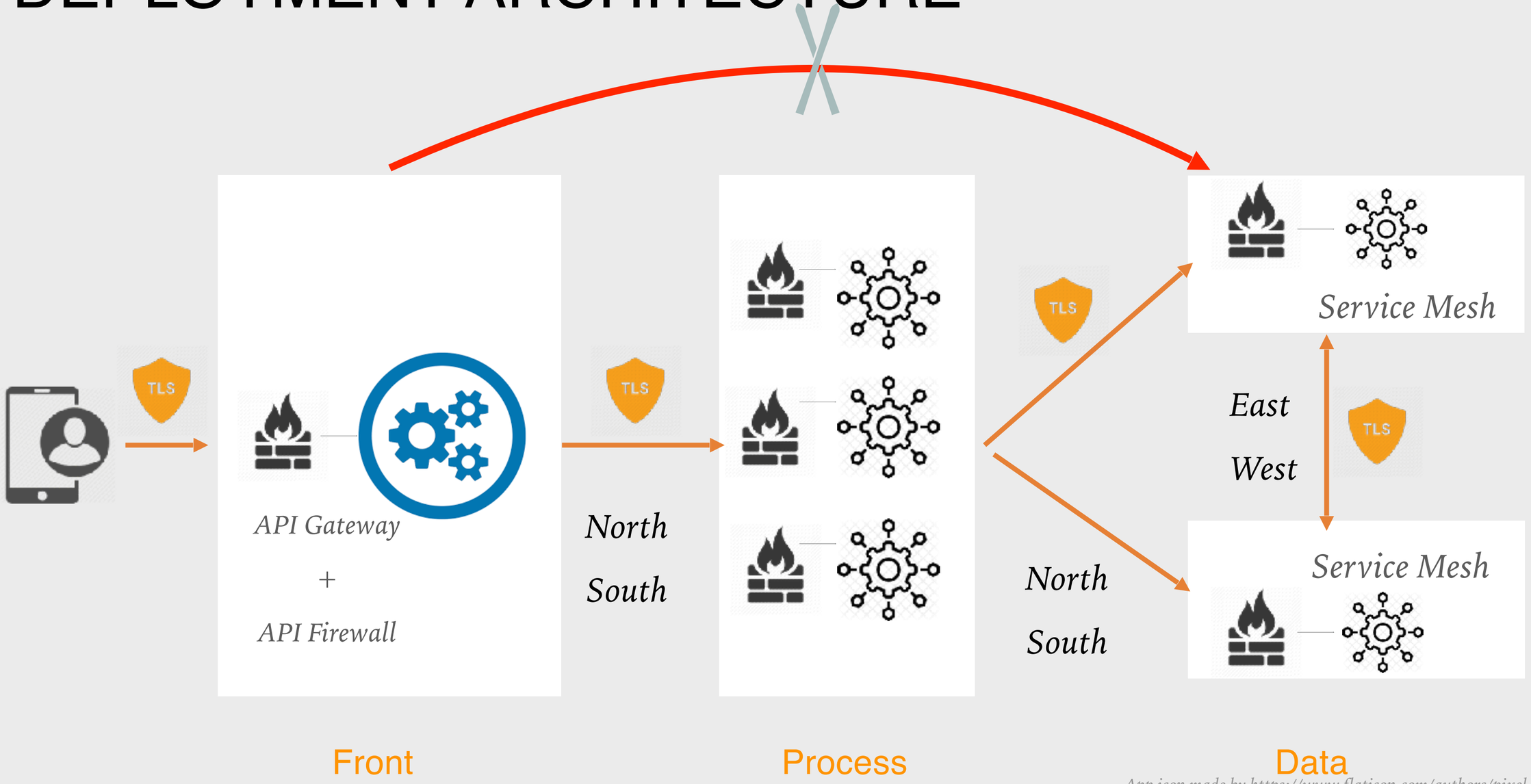
App level security (Auth, Azn, libs, code, images, data)

Intra-services communication (Auth, Azn, TLS)

Hypervisor, Kubernetes

OS / Network / Physical Access

DEPLOYMENT ARCHITECTURE



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

{api addicts}

References

- **Input Validation**
  - https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
- **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

- **Mitigate OWASP Top10 with API Management**
  - https://learn.microsoft.com/en-us/azure/api-management/mitigate-owasp-api-threats
- **42Crunch webinars with Jim Manico on CSRF :**
  - https://42crunch.com/defending-apis-with-jim-manico-episode-1/

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}