{api addicts}

/ SEGURIDAD EN LAS APIS

{escuela de APIs}

{api addicts}

@isamauny

- Chief Technology Officer and co-founder @42Crunch
- 42Crunch is the company behind apisecurity.io
- Working with APIs since 2005!
- Most career at IBM
- Command-line fan
- French native, in Spain since 2003

isabelle@42crunch.com

API Breaches are on the rise!

- 300+ breaches reported on apisecurity.io since Oct. 2018
- And those are just the public ones!
- Most recurrent causes (combination of):
  - Lack of Input validation
  - Lack of Rate Limiting
  - Data/Exception leakage
  - Authorization issues
  - Authentication issues

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

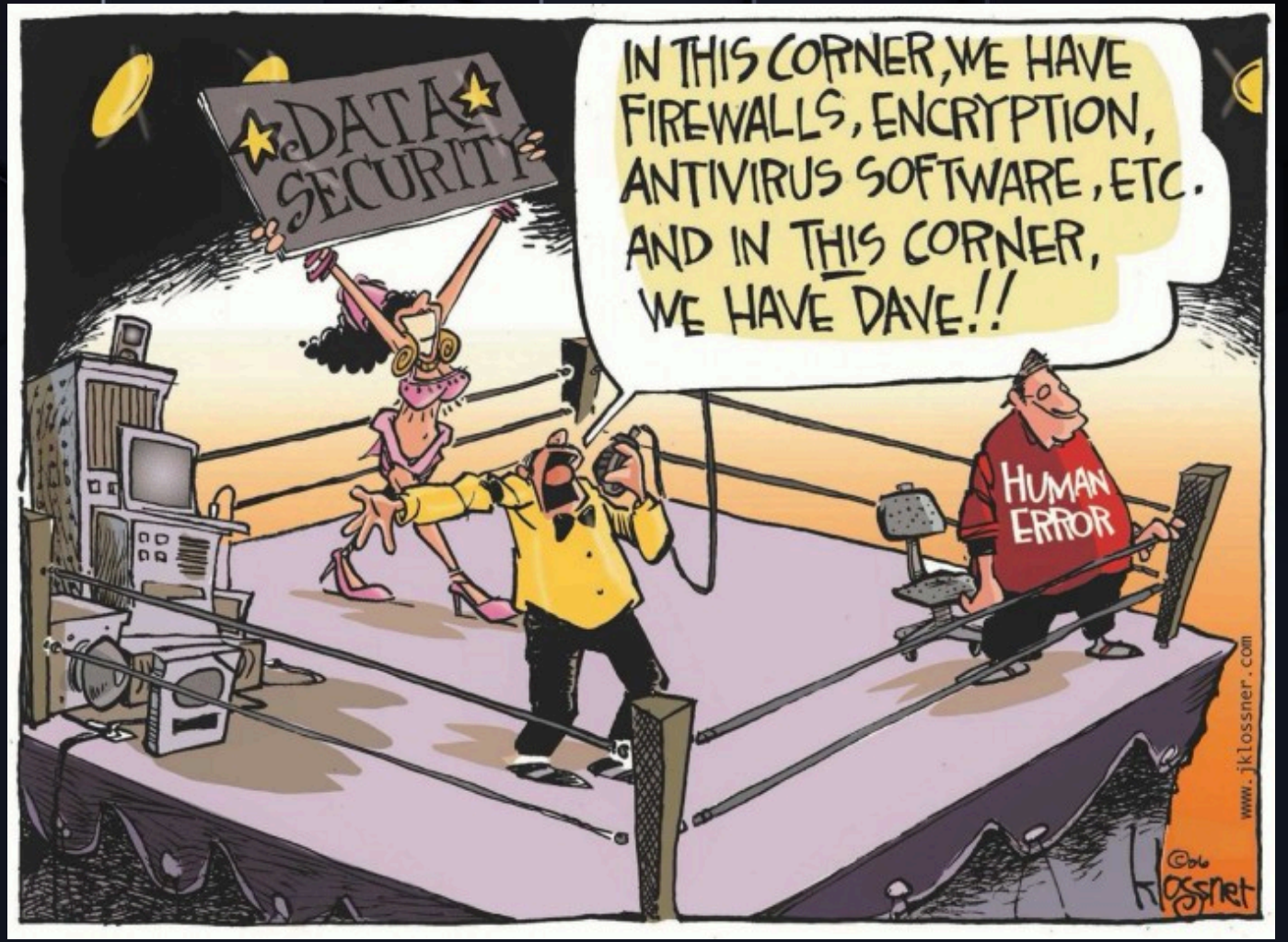
Equifax - 147 million users personal data stolen

Starbucks - 100 million customer records accessed

Verizon - 14 million subscribers phone numbers and PINs exposed

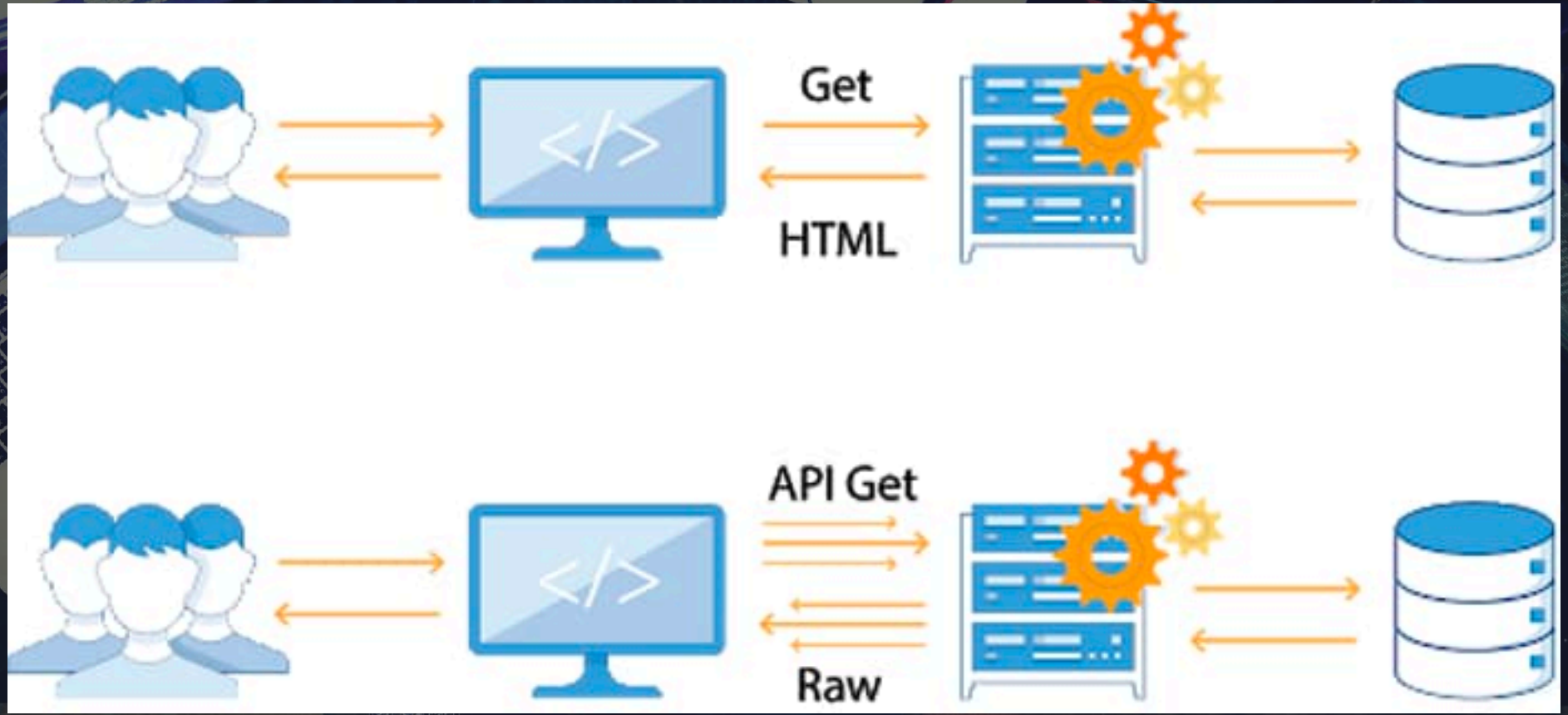
WHY IS THIS HAPPENING?

WE ARE HUMANS!



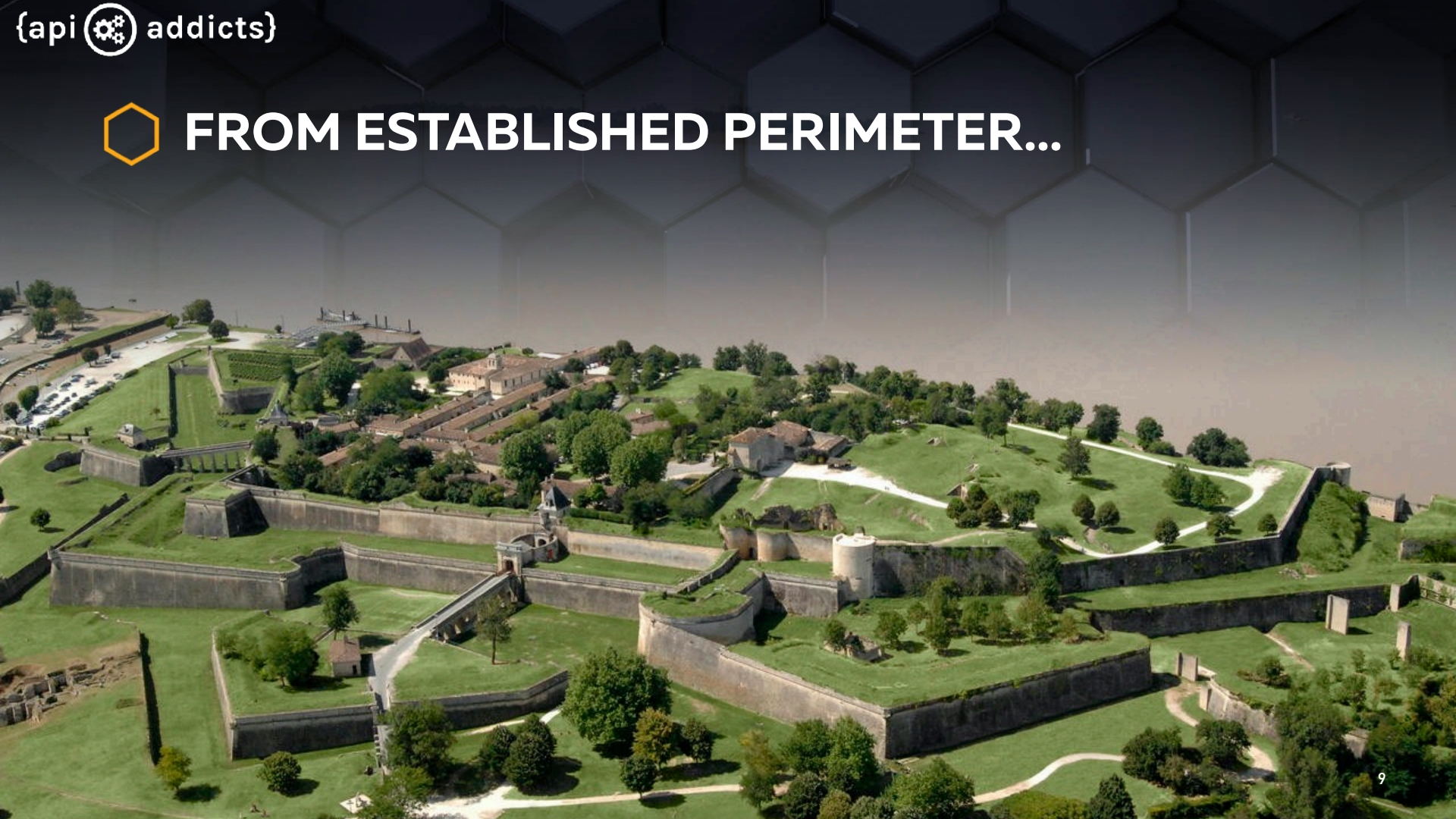

For the security we said we'd hire someone to...
Hold on, that's not the priority. Let's get the project started first, we'll see about that later...
Hey the project is nearly done, we should do a security audit to...
Too late man, we don't have the time or the budget for that
The site's been online for 10 days, and we haven't done any security testing, I'm nervous...
Don't worry, we'll take care of it later...
LATER...
DOWN! THE SITE IS DOWN! WE'VE BEEN HACKED!
How could you let this happen!?
CommitStrip.com

Applications Architecture has changed!



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

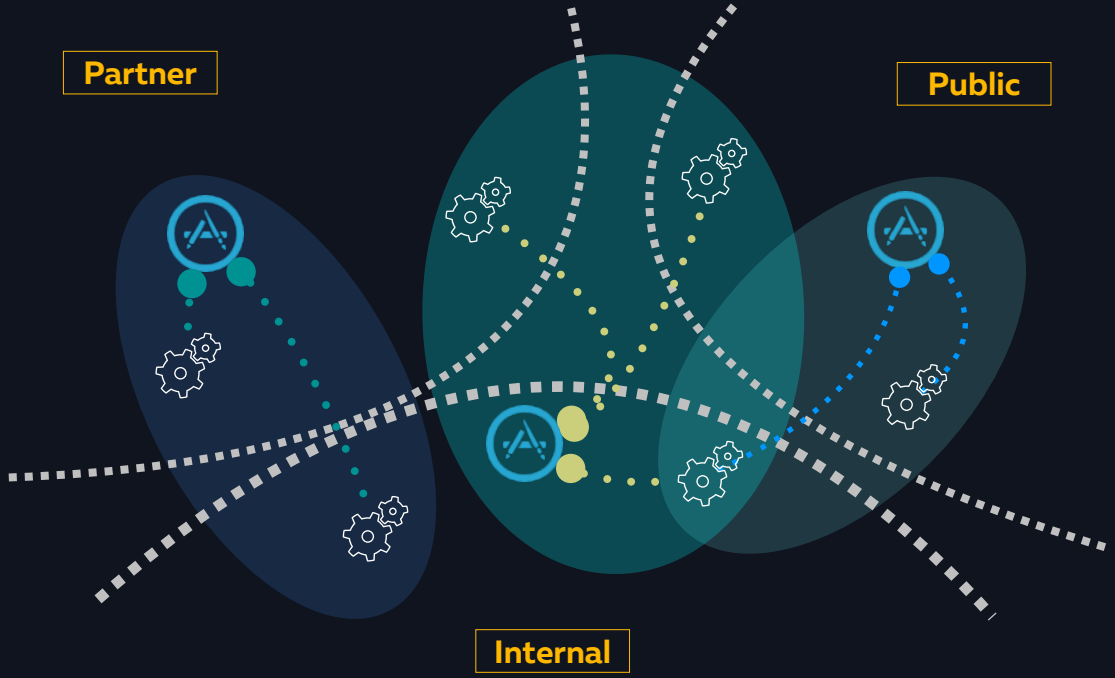
FROM ESTABLISHED PERIMETER...



{api addicts}

...TO BLURRY PERIMETER

VIRTUAL APPLICATION NETWORKS



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

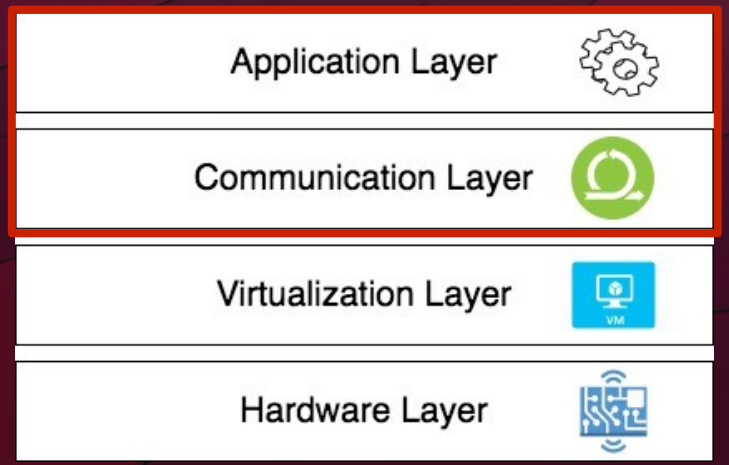
TOO MANY APIS, TOO MANY DEPLOYMENTS

APPLICATION DEVELOPMENT

APPLICATION SECURITY

DEFINING API SECURITY

{api addicts}



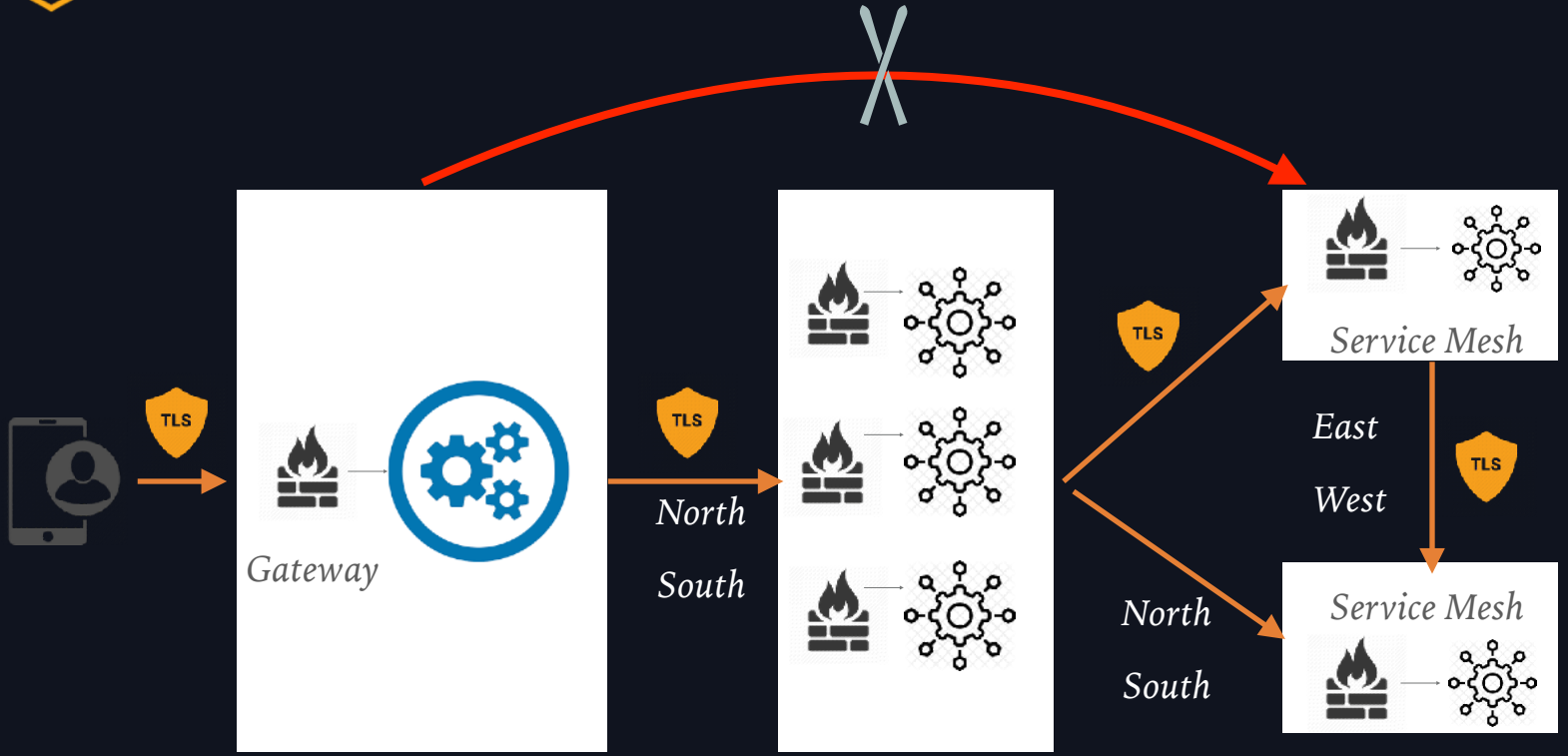
App level security (auth, azn, libs, code, data)

Intra-services communication (auth, azn, TLS)

Hypervisor, images (VM/Docker)

OS / Network / Physical Access

NEW DEPLOYMENT ARCHITECTURES



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

APP SEC REQUIREMENTS

Input/Output Validation
(Attacks Protection)

Integrity
(Data has not been tampered with)

Confidentiality
(Data can't be seen in flight)

Availability
(Rate Limiting)

Authentication
(Validation and OIDC Flows)

Authorization
(Access Control and OAuth flows)

Audit
(Forensics)

Non Repudiation
(Legal Compliance)

ALL APIS SHOULD BE TREATED AS PUBLIC

YOU CAN'T PROTECT WHAT YOU DON'T KNOW.

“I think that a lot of people think that because there is **no GUI** on an API that **no one can find it** and it is invisible. **But we can find them in about five seconds** with a proxy…

…**Almost every threat that applies to a web app, can happen to an API**, but a lot of people for some reason are not protecting them as much as their web applications.”

Tanya Janca

Application Security Evangelist - AppSec Podcast

WHAT SHOULD YOU DO ?

- Proceed to a full inventory of APIs within the enterprise
- Implement APIs governance
- Evaluate your API Security coverage

SECURITY NEEDS TO BE RISK-BASED

“Security is a risk control measure…**In the security sphere, one size does not fit all**. We have to take ‘appropriate measures’.

Nat Sakimura - OAuth Creator

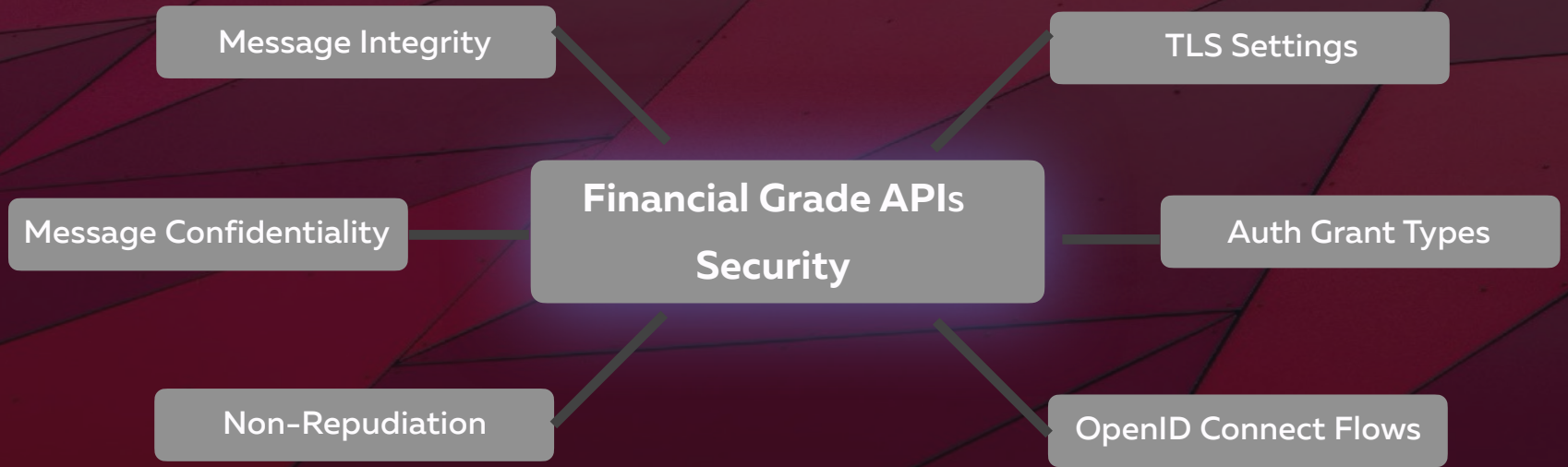
Fixing OAuth, Nat Sakimura, July 20, 2016, https://nat.sakimura.org/2016/07/20/fixing-oauth/

WHAT SHOULD YOU DO ?

- Establish a threat model for all APIs
- Establish corporate security policies based on that threat model, managed by the security teams.

OPEN BANKING SAMPLE CATEGORISATION

API	Category	Data Sensitivity	Operations Sensitivity	Authentication Requirements	Integrity Requirement
ATMs	OpenData	Low	Low	None	None
Accounts	Read	High	Medium	Medium	Required
Payments	Write	High	High	High	Required


Financial Grade APIs Security
Message Integrity
Message Confidentiality
Non-Repudiation
TLS Settings
Auth Grant Types
OpenID Connect Flows

SECURITY MUST BE AUTOMATED

LET'S SHIFT LEFT!

https://code.likeagirl.io/pushing-left-like-a-boss-part-1-80f1f007da95

OWASP API SECURITY TOP 10

FINALLY! OWASP API SECURITY TOP 10

- A recognition that API Security is **not** Web Application Security
  - ✓ Different attack vectors
  - ✓ Data centric (lots of attacks come from mishandling API data)

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

{api addicts}

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

OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

MEET PIXI !

- An API that exposes all the issues APIs can have!
- Totally vulnerable!
- Part of an OWASP Project called DevSlop

OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

A1: OBJECT LEVEL ACCESS CONTROL

EXTRACTION VIA ENUMERATION

- APIs relies on public IDs to retrieve information, for example
  - ✓ GET /shops/**shop1**/revenue_data.json
  - ✓ Returns info from **shop1** if you're authenticated
  - ✓ What happens if I call GET /shops/**shop2**/revenue_data.json?
- In 2017, a major flaw was discovered in a T-mobile API which would have allowed to retrieve any T-mobile customer information from their phone number. All you needed was a valid token.
- What can you do ?
  - ✓ Implement authorization checks to validate the user actually owns the data they are trying to retrieve
  - ✓ Use non-guessable IDs (UUIDs)
  - ✓ Don't expose internal IDs externally

UBER (SEPTEMBER 2019)

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

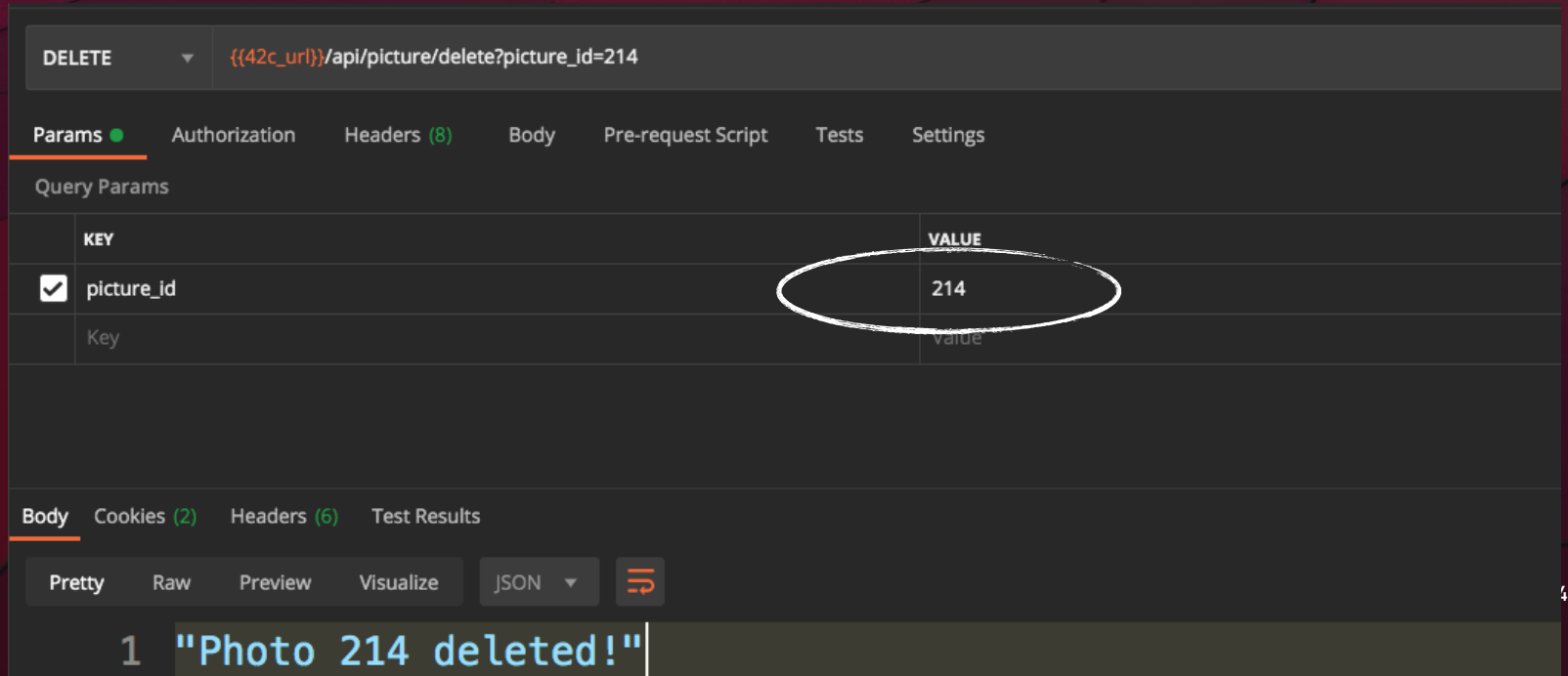
- The Attack
  - ✓ Account takeover for any Uber account from a phone number
- The Breach
  - ✓ None. This was a bug bounty.
- Core Issues
  - ✓ First Data leakage : driver internal UUID exposed through error message!

```
{
    “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
  - ✓ Second Data leakage via the getConsentScreenDetails operation: 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

A1 AND PIXI

- Can delete someone else pictures!
- DELETE {{42c_url}}/api/picture/delete?picture_id = 124



OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

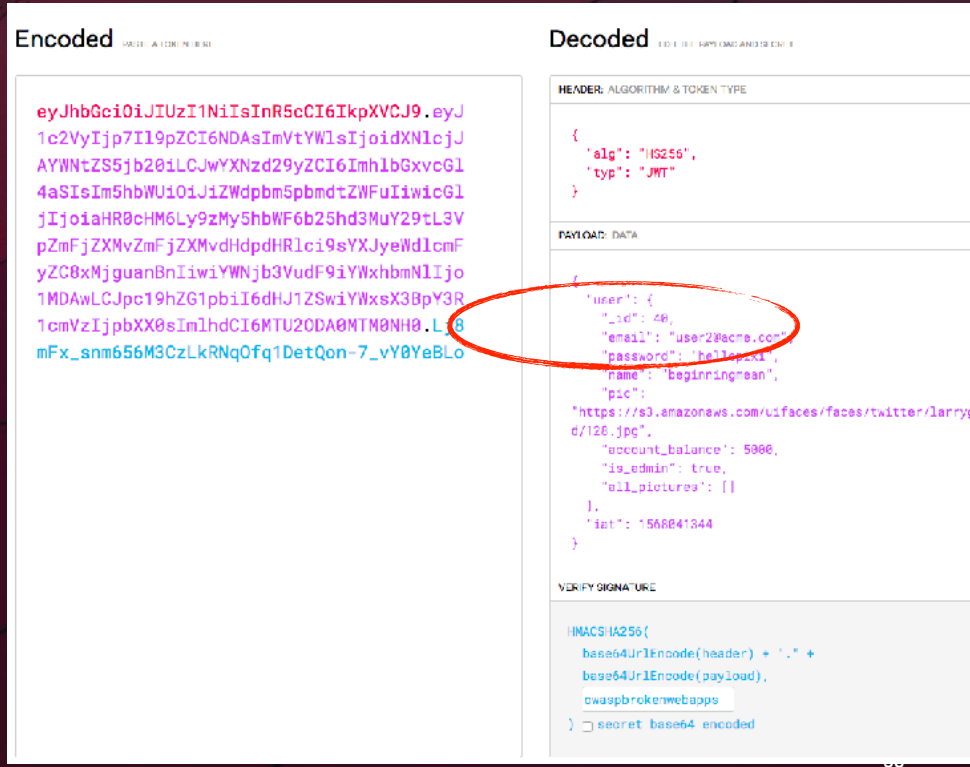
A2:BROKEN AUTHENTICATION

POOR AUTHENTICATION AND ACCESS TOKENS MISUSE

- Weak authentication: BasicAuth, static APIKeys.
- A single token allows to access many APIs and operations
- Poorly protected API Keys and OAuth secrets: they are leaking all over Github and Amazon S3
  - ✓ A very easy mistake! Check Github Token Scanning.
- What you should do
  - ✓ Restrict access token usage
    - *Google APIs example (restrictions on referrers, APIs list, IPs).*
    - *OAuth scopes*
    - *Fine-grained authorization (attribute-based for example)*
  - ✓ Use short-lived access tokens
  - ✓ Authenticate your apps (check who has the right to call your APIs)
- Remember: OAuth is NOT an authentication protocol!

A2 AND PIXI

- Let's forge a JWT token (I know the secret, see A7!)
- Specify an ID which is the user identifier and you can get the user information !



OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

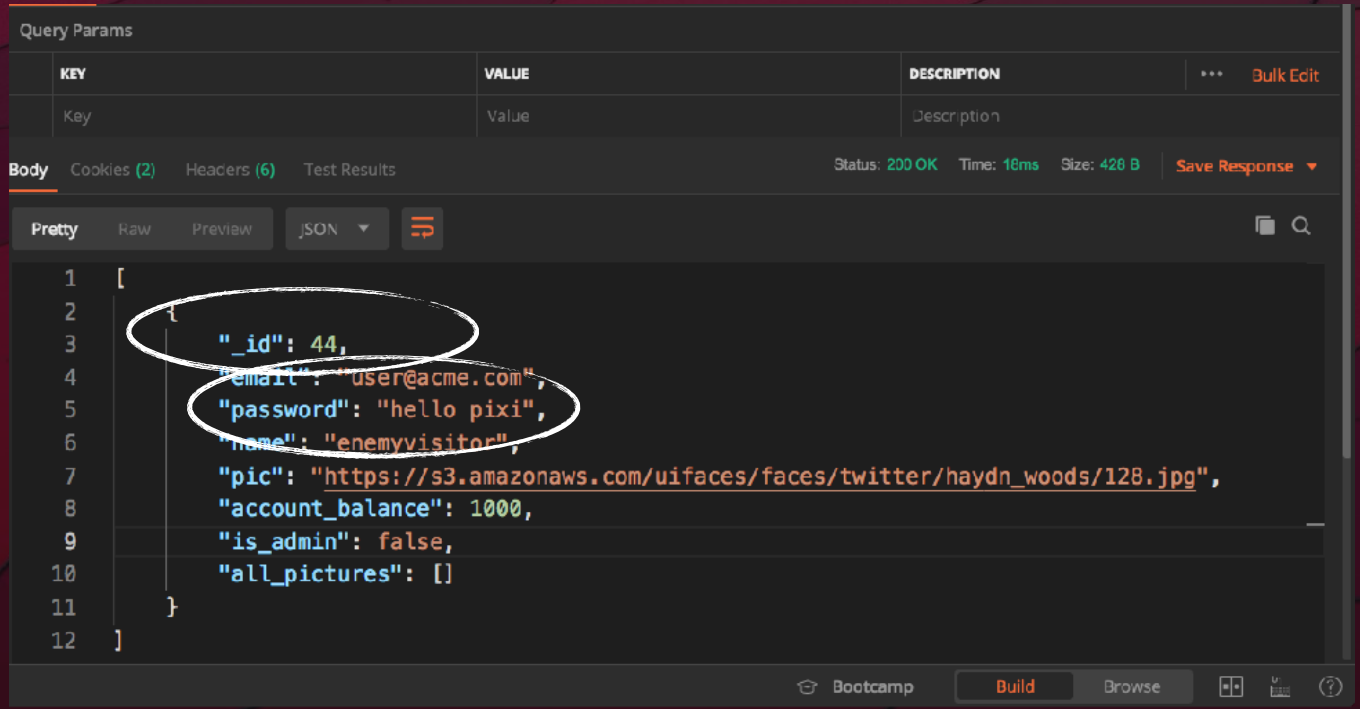
A3: EXCESSIVE DATA EXPOSURE

PREVENT DATA LEAKAGE

- Developers tend to create generic APIs which expose lots of information and let client UIs filter what they need.
- APIs return excessive information
- APIs leak exception information (exposing server internals)
- What you can do
  - ✓ Describe precisely the response structure and block anything not matching the specification
  - ✓ Control the data an API can expose externally

A3 AND PIXI

- GET {{42c_url}}/api/user/info



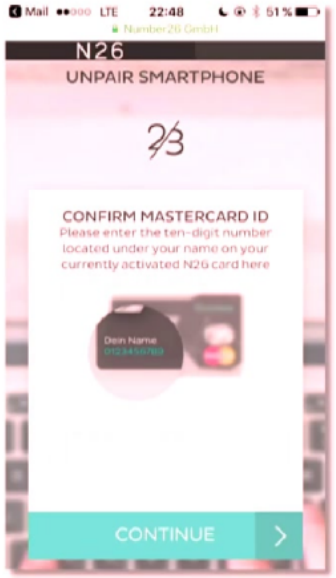
N26 EXAMPLE

Unpair: MasterCard ID

- MasterCard ID is printed on the card
- However, each transaction contains the following:

```
{
  "amount": -0.11,
  "cardId": "b8484ca2-a674-4f1c-afd1-896a3bfe6d15",
  "linkId": "0123456789-372287",
  "merchantCity": "DUESSELDORF  ",
  "merchantCountry": 0,
}
```

MasterCard ID is part of every MasterCard transaction!



#sec26

27

OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

A4: LACK OF RESOURCES AND RATE LIMITING

LIMIT ACCESS TO APIS

- APIs are subject to abuse (brute force attacks for example) if calls rate is not limited properly
- DOS type attacks exhaust system resources
- Things to watch for
  - ✓ Don't base rate limiting just on IP address : not sufficient as cloud platforms can be used by hackers to spawn thousands of instances that all have different IPs.
  - ✓ Rate limiting should use request information to count requests, for example user identity inside a JWT or APIKey passed in a header
  - ✓ Use intervals to limit how quickly calls can be made (i.e. leave 150 ms across 2 calls from same identified client)
  - ✓ Limit resources an API server can use: for example, in Docker/ K8s use CPU/Mem/threads limits.

A MORE INTERESTING SAMPLE THAN PIXI

- Brute force attack on Instagram to guess 6-digit password reset code
- Rate limiting of 200 attempts / minute ! per **IP**
- Hackers circumvented this by renting AWS machines, all with different IPs.

OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

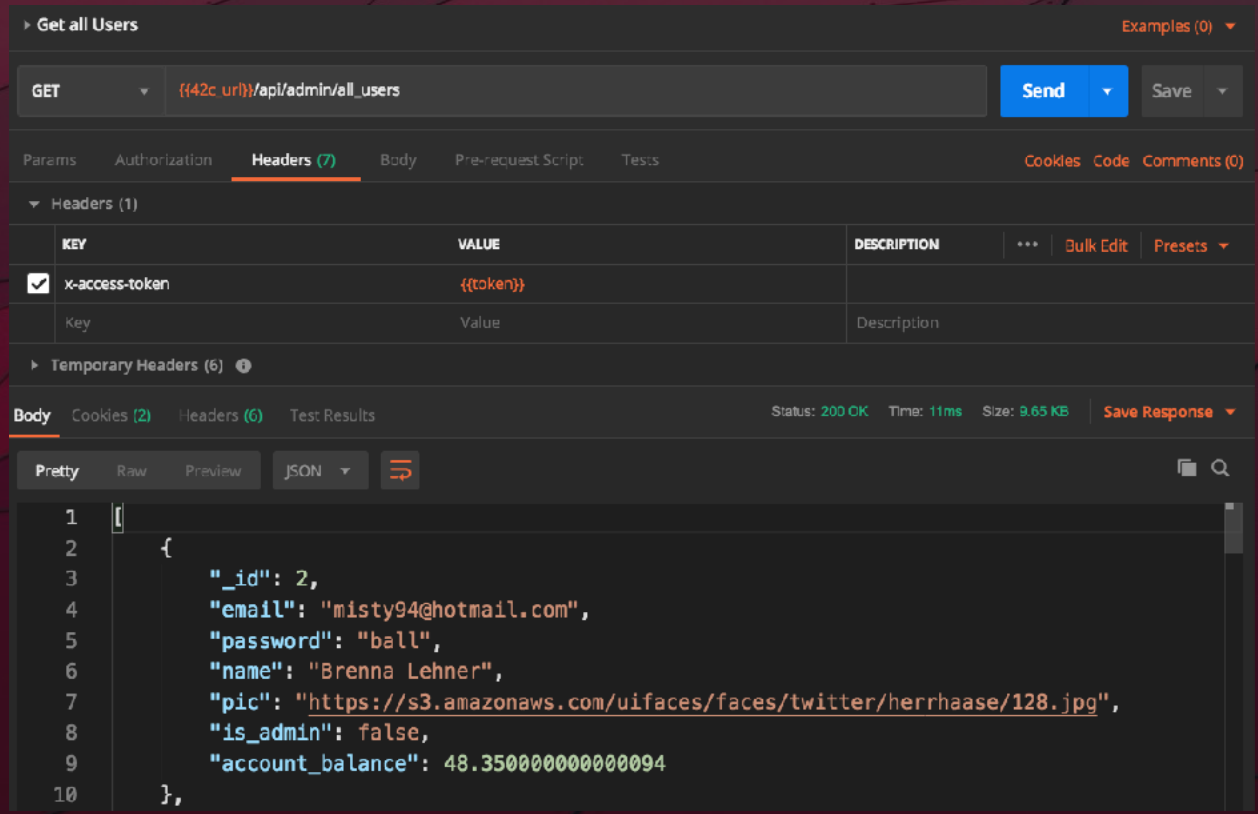
A5: RESOURCE LEVEL ACCESS CONTROL

BLOCKING UN-AUTHORIZED CALLS

- Example:
  - ✓ Developers mix admin operation together with non-admin ones in same API
  - ✓ Developers leave sensitive verbs (PATCH/DELETE) in API implementation that can be exploited to update/delete data with no further authorization
- Hackers use bots to guess which potential paths are open (say GET /admin/user or PUT /user) and get access to information that should be hidden.
- What you can do
  - ✓ Detect whether verbs not specified in API contract can be used
  - ✓ Systematically reject any path not described in the API contract
  - ✓ Validate that the user has proper access rights

A5 AND PIXI

- Administrative operations blocked via *app* checks (not API checks)



OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

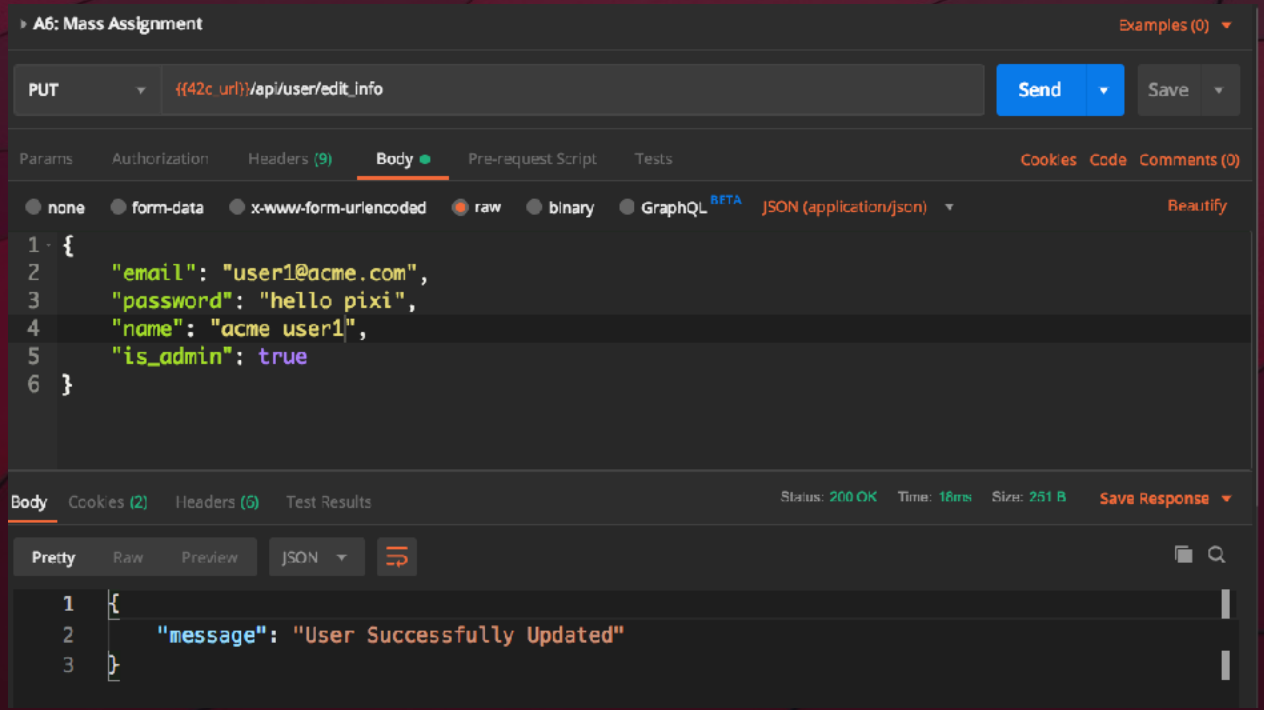
A6: MASS ASSIGNMENT

CONTROLLING DATA UPDATES

- APIs support changing data via mass assignment, leading for example to an hacker updating their role or updating a resource
- Related to A3 (exposing too much data)
- What you can do
  - ✓ OpenAPI contract describes exactly which information can be passed for each operation/verb
  - ✓ Systematically rejects any request not conforming to the contract

A6 AND PIXI

- Making myself an admin!



A7: MISCONFIGURED SECURITY

SECURITY MISCONFIGURATION

- Known API breaches:
  - ✓ Equifax breach was due to a crafted Content-Type header - This Apache Struts vulnerability had been reported and fixed, but not deployed at Equifax.
- Misconfigured environment
  - ✓ Security is only as strong as its weakest link!
- Detect vulnerable libraries
  - ✓ Node.js!! Solutions like Snyk and Github tools.
- What you can do
  - ✓ Limit external dependencies
  - ✓ Continuously assess system security configuration
  - ✓ Use known and trusted libraries!
  - ✓ Don’t code your own security stack …

A7 AND PIXI

- Direct access to server.conf file
  - ✓ http://localhost:8000/server.conf
  - ✓ Leaks session secret! now we can create JWTs!
- Unprotected MongoDB admin console
  - ✓ http://localhost:28017

USING NODE.JS ? READ THIS !

In some wise words from Google:

> *If an attacker successfully injects any code at all, it's pretty much game over*

XSS is too small scale, and really well protected against.

Chrome Extensions are too locked down.

Lucky for me, we live in an age where people install npm packages like they're popping pain killers.

★ Top highlight

https://hackernoon.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5

Malicious code found in npm package event-stream downloaded 8 million times in the past 2.5 months

NOVEMBER 26, 2018 | IN VULNERABILITIES | BY DANNY GRANDER

A widely used npm package, `event-stream`, has been found to contain a malicious package named `flatmap-stream`. This was disclosed via a GitHub issue raised against the source repo.

The event-stream package makes creating and working with streams easy, and is very popular, getting roughly 2 million downloads a week. The malicious child package has been downloaded nearly 8 million times since its inclusion back in September 2018.

We have added the malicious package to our vulnerability database. If your project is being monitored by Snyk and we find the malicious dependency (either `event-stream@3.3.6` or any version of `flatmap-stream`) you will be notified via Snyk's routine alerts.

APPLIES TO DOCKER IMAGES TOO!

July 22, 2020

Anatomy of a Kubernetes Attack - How Untrusted Docker Images Fail Us

PART TWO OF A SERIES.

This post illustrates how an attacker could use a poisoned docker image to break out of a container and gain access to the hosts\nodes in a Kubernetes cluster.

In the first part of this blog series, Infrastructure as Code: Terraform, AWS EKS, Gitlab & Prisma Cloud, I went through the scenario of building out an Amazon EKS cluster using Terraform and a VCS integration with Gitlab. I also covered an initial Daemonset deployment of the Palo Alto Networks Prisma agent to the nodes on the newly provisioned cluster. In the second part of this series, we will look at some aspects of a Kubernetes attack and in a follow-up installment, how Palo Alto Networks Prisma Compute can be used to prevent related Kubernetes attacks.

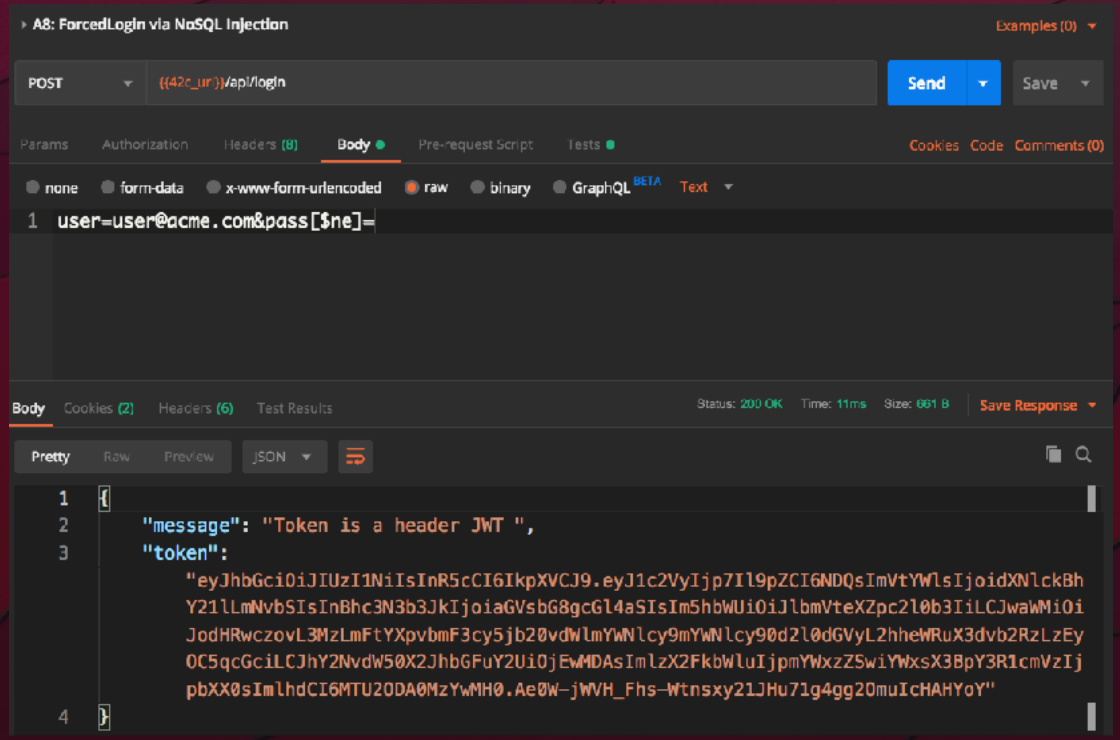
Microsoft recently released a Kubernetes attack matrix (similar to MITRE's ATT&CK framework) that I feel is a good reference for walking through some of the aspects of an example Kubernetes attack. For this blog, I'll be covering techniques in Initial Access, Execution, Persistence and Privilege Escalation Tactic phases.

PREVENTING TYPICAL INJECTIONS

- This is a common issue across web applications and APIs.
- Remote code injection, SQL injection, noSQL injection all fall in this category
- Samsung Smarthing is a good example of SQL Injection leading to crashing the video server
- What you need to do:
  - ✓ Sanitize inputs

A8 AND PIXI

- NoSQL MongoDb injection



OWASP

OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

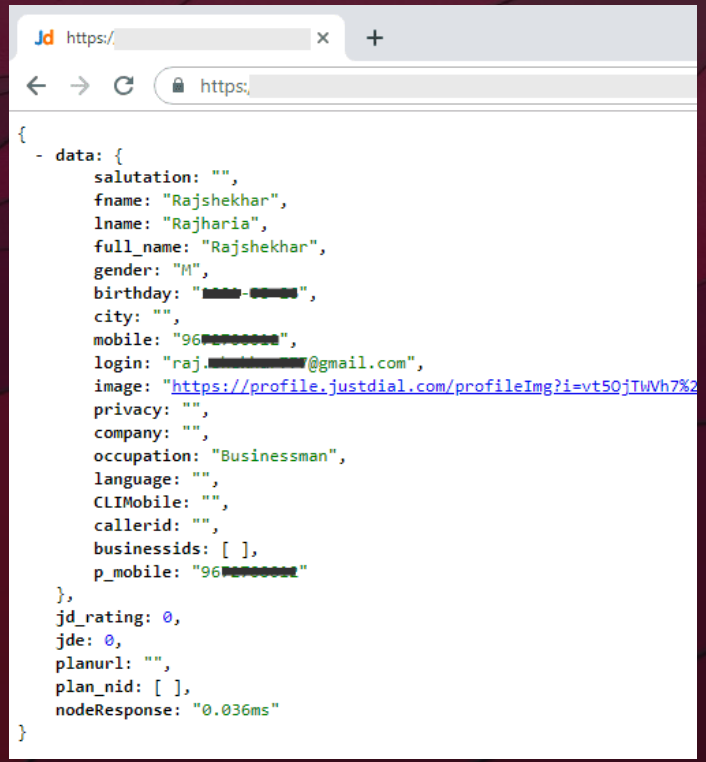
A9:IMPROPER ASSETS MANAGEMENT

IMPROPER ASSETS MANAGEMENT

- Document all APIs with OpenAPI/RAML
- Implement APIs governance
- Make sure API versions are tracked (deprecated, retired, etc.)
- Separate production from non-production traffic
- API deployment can be blocked via CI/CD pipelines to prevent unsecured APIs from being deployed, even in Dev/QA.
- Adopt DevSecOps so that security is considered as early as possible. This way, even Dev APIs are protected.

JUSTDIAL (2019)

https://apisecurity.io/issue-28-breaches-tchap-shopify-justdial/



- India's largest local search service
- Had old unused unprotected API
- It was connected to live database
- Was leaking 100 millions users' data

OWASP

OWASP API Security Top 10 2019

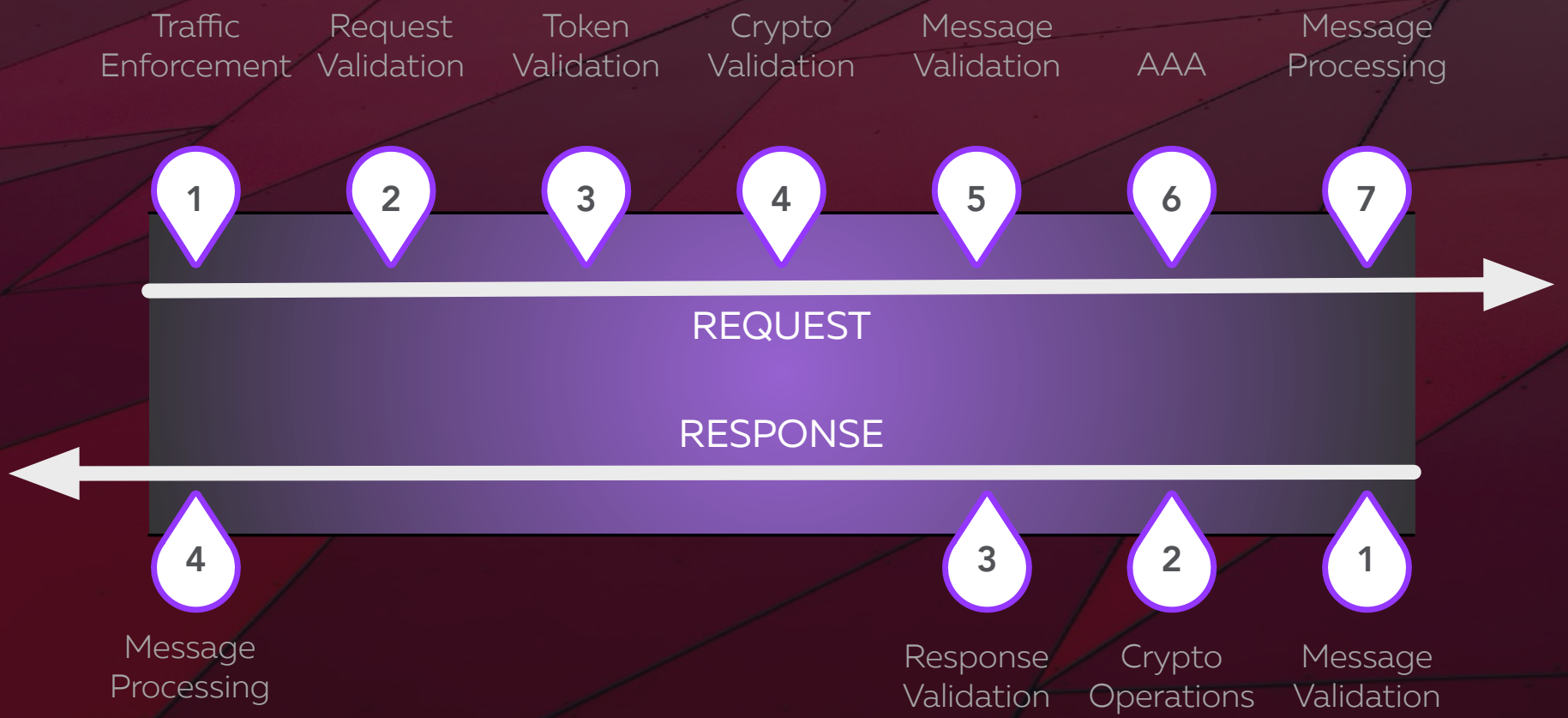
The Ten Most Critical Web Application Security Risks

A10: INSUFFICIENT LOGGING AND MONITORING

API SECURITY VISIBILITY

- Abuse of APIs goes un-noticed … i.e. you realize weeks later that you had a breach.
- Produce transaction logs which can be exported to 3rd party systems, such as SIEMs for further analysis and exploitation
- Protect Logs !
  - ✓ Make sure you're not vulnerable to log injection!
- Watch what you write in logs!
  - ✓ Token and keys passed in URL versus HTTP Headers
  - ✓ Tokens and Keys written from code
  - ✓ Use Redaction whenever possible

API SECURITY BLUEPRINT


Traffic Enforcement
Request Validation
Token Validation
Crypto Validation
Message Validation
AAA
Message Processing
1
2
3
4
5
6
7
REQUEST
RESPONSE
4
3
2
1
Message Processing
Response Validation
Crypto Operations
Message Validation

IT ALL STARTS WITH TRANSPORT

- TLS covers Confidentiality and Integrity **at transport** level.
- Configuration matters!
  - ✓ Protocol accepted (TLS 1.2, 1.3 are recommended)
  - ✓ Cipher suites
- Can use Mutual TLS for authentication is some scenarios
- Review/Enforce across the whole transaction flow
  - ✓ Inbound/Outbound
- Remember: channel is encrypted… but data goes in clear!

VERIFYING SSL/TLS SETUP

- www.ssllabs.com
- report.io
- securityheaders.io

REQUEST VALIDATION

- Verbs
- Path
- Headers
- Query params
- Cookies
- CORS
- Apply positive and negative security models (a.k.a whitelisting and blacklisting)
- Leverage Open API to apply positive security model!

MESSAGE VALIDATION

- Payload validation (request, responses, errors!)
- Block sensitive data in responses (N26 attack lessons…)
- Make sure you don’t return too much information in case of errors. Too much info for attacker!
  - ✓ Avoid Response.post (exception.printStackTrace) !

AUTHENTICATION/AUTHORISATION

OAUTH THREAT LANDSCAPE

- Sensitive information
  - ✓ Access tokens: the key to open the API door
  - ✓ Refresh tokens the key to obtain many valid access tokens
  - ✓ Authorization code : to obtain an access token
  - ✓ State : protection against CSRF attacks
- Common attacks vectors
  - ✓ **Token leakage** → TLS 1.2 minimum, preferably 1.3 + strong ciphers, short-lived, token binding, protected storage
  - ✓ **Code Interception attacks** → Use PKCE (Proof Key for Code Exchange)
  - ✓ **Token replay** → Token Binding (over TLS) or Mutual TLS for OAuth
  - ✓ **Redirect URI Forgery** → standard CRSF attacks countermeasures (OWASP)

REQUIRES PROTECTION MEASURES ON CLIENT AND SERVER SIDE

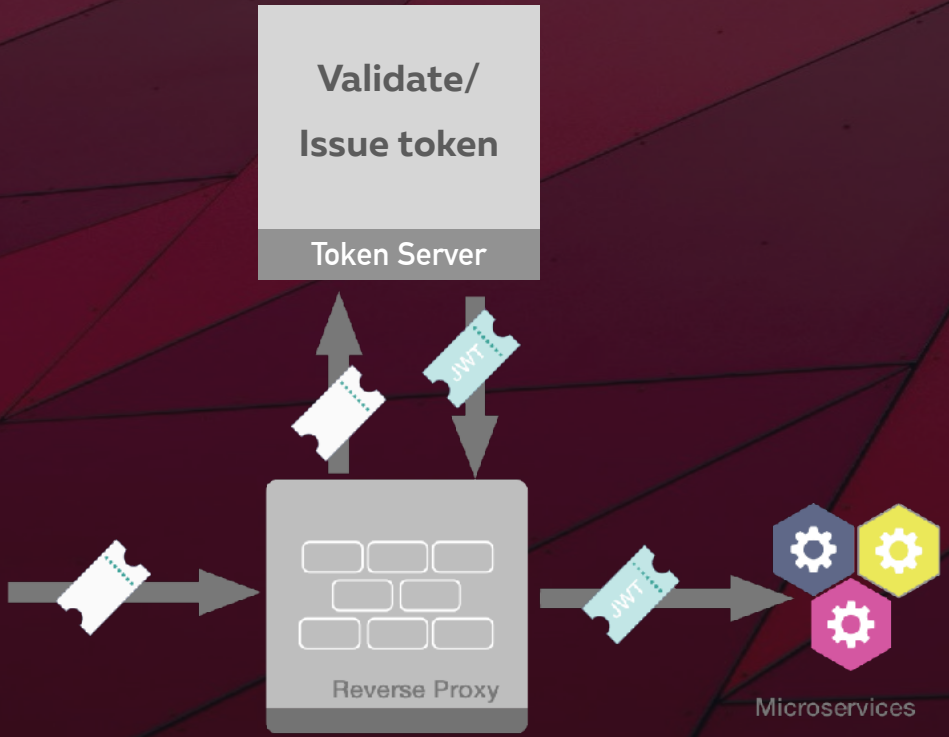
ACCESS TOKENS AND API KEYS

- Prefer tokens to API keys
  - ✓ API keys do not expire, tokens do
  - ✓ Tokens embark information, API keys are usually opaque
- How do we obtain a token ?
  - ✓ Generally OAuth
- What the token format ?
  - ✓ Usually JWT (JSON Web token)
- OAuth Grant Types you should use
  - ✓ Authorization Code
  - ✓ Others are usually not the safe option.
- An access token **is not a** proof of authentication, use OpenID Connect instead.

OAUTH CHEAT SHEET

- Use HTTPs across all actors (Resource Server, Authorization Server, Client)
- Prevent Token theft ! Look at
  - ✓PKCE (prevents authorization_code from being stolen)
  - ✓Proof-of-possession (https://tools.ietf.org/html/rfc7800)
  - ✓Token Binding (new RFC)
- Use proven libs and products !
- Learn Learn and Learn …
  - ✓https://auth0.com/docs/api-auth/grant/authorization-code-pkce
  - ✓https://alexbilbie.com/guide-to-oauth-2-grants/
  - ✓https://medium.com/@robert.broeckelmann/when-to-use-which-oauth2-grants-and-oidc-flows-ec6a5c00d864

EXTERNAL VS INTERNAL TOKENS



{api addicts}

VALIDATING TOKENS

- Don't trust the incoming tokens!
  - ✓Applies to North-South, East-West traffic
- Client could have been compromised...
- Validate algorithm (the one you chose!)
  - ✓ HS256/384/512
  - ✓ RS256/384/512
  - ✓ PS256/384/512 (recommended)
- Reject None (and none, NonE or nonE ....)
- Validate signature
  - ✓ Prefer digital signatures over HMAC
  - ✓ If not, be careful of key exchange
- Validate standard claims and your own claims
- Check Audience
- Check Expiration
- See details Learn the best practices for keeping your JWTs secure.


Paul Reinheimer
@preinheimer
Follow
The more I learn about cryptography, the more I think Alice and Bob should probably just talk in person.
6:01 AM - 13 Mar 2017
3,639 Retweets 6,059 Likes
83
3.6K
6.1K

CRYPTO VALIDATION

- Can I decrypt ?
- Can I verify the signature ?
- Decrypt before payload validation !

INTEGRITY

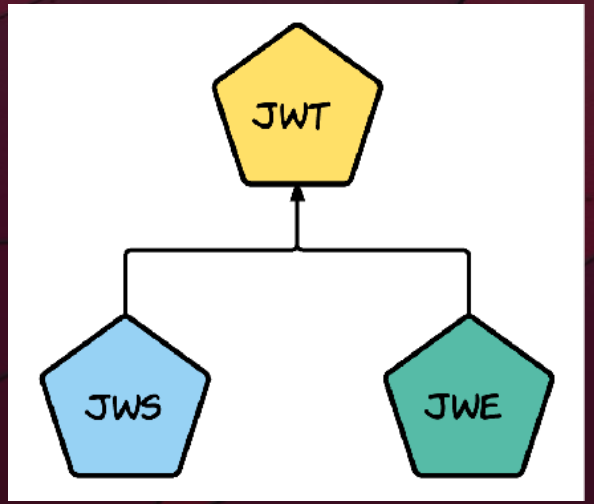
- What I received is what was sent and I know **who** sent it.
- Digital signatures over content.
- You may already use this with OpenID Connect (id token must be signed and optionally encrypted)
- Transport agnostic!
- Other applications
  - ✓ Non-Repudiation

CONFIDENTIALITY

- I don’t want anybody to see the messages exchanged.
- Data can only be read by the right person/system
- Transport agnostic!
- Multiple recipients
  - ✓ Part of message goes to target A, another to target B

USEFUL ACRONYMS

- JOSE: Javascript Object Signing and Encryption
  - ✓ IETF Standard for JWS and JWE
- JWE
  - ✓ JSON Web Encryption
- JWS
  - ✓ JSON Web Signature
- JWT
  - ✓ JSON Web Token
- JWK
  - ✓ JSON Web Key



RESOURCES

- Chaos Engineering
  - ✓ http://principlesofchaos.org
  - ✓ https://github.com/dastergon/awesome-chaos-engineering
- OWASP ZAP
  - ✓ https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
- Source Code Analysis
  - ✓ https://www.owasp.org/index.php/Source_Code_Analysis_Tools
- Code Security reviews
  - ✓ https://www.owasp.org/index.php/Code_Review_Introduction
- Systems Scans
  - ✓ https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools
- Security Methodology
  - ✓ https://developer.rackspace.com/blog/fanatical-security-delivered-by-quality-engineering-security-team/

RESOURCES

- SSL Setup Scan
  - ✓ https://hardenize.com
  - ✓ https://securityheaders.io
  - ✓ https://www.ssllabs.com/ssltest/
- Threat Modelling
  - ✓ https://www.owasp.org/index.php/Application_Threat_Modeling
- Attacks Type Information
  - ✓ XSS: https://excess-xss.com
  - ✓ Buffer Overflow: https://www.youtube.com/watch?v=1S0aBV-Waeo
  - ✓ SQL injection: https://www.youtube.com/watch?v=ciNHn38EyRc
  - ✓ Cookie stealing /XSS: https://www.youtube.com/watch?v=T1QEs3mdJoc
- Pixi / DevSlop
  - ✓ https://github.com/DevSlop/Pixi
  - ✓ https://devslop.co
- JWT as session data
  - ✓ https://dzone.com/articles/stop-using-jwts-as-session-tokens
- Node JS Security recommendations
  - ✓ https://medium.com/@nodepractices/were-under-attack-23-node-js-security-best-practices-e33c146cb87d

{api addicts}

ApiAddicts te agradece por asistir

/CloudAPPi