API Security Project

OWASP Projects' Showcase

Sep 12, 2019

Founders and Sponsors

Project Leaders

Erez Yalon

- Director of Security Research @ Checkmarx

- Focusing on Application Security

- Strong believer in spreading security awareness

Inon Shkedy
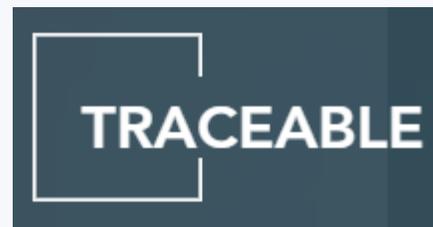
- Head of Research @ Traceable.ai

- 7 Years of research and pentesting experience

- I've grown up with APIs

What is API?

API stands for:

Application **P**rogramming **I**nterface

"An application programming interface (API) is ==an interface or communication protocol between a client and a server== intended to simplify the building of client-side software. It has been described as a "contract" between the client and the server, such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action."

https://en.wikipedia.org/wiki/Application_programming_interface

Who Uses APIs?

Every Modern application:

- Mobile

- IoT

- B2B

- Serverless

- Cloud

- Single Page Application

We

API

API Security
==
API-Based Apps Security

Today's Agenda

- How APIs-Based apps are different? Why deserve their own project?
- Roadmap
- **API Security Top 10 RC**
- Acknowledgements
- Call for contributors

# How API Based Apps are Different?

Client devices are becoming varied and stronger

Logic moves from Backend to Frontend
(together with some vulnerabilities)

Traditional vs. Modern

Traditional Application

Get

HTML

Modern Application

API Get

Raw

elastic
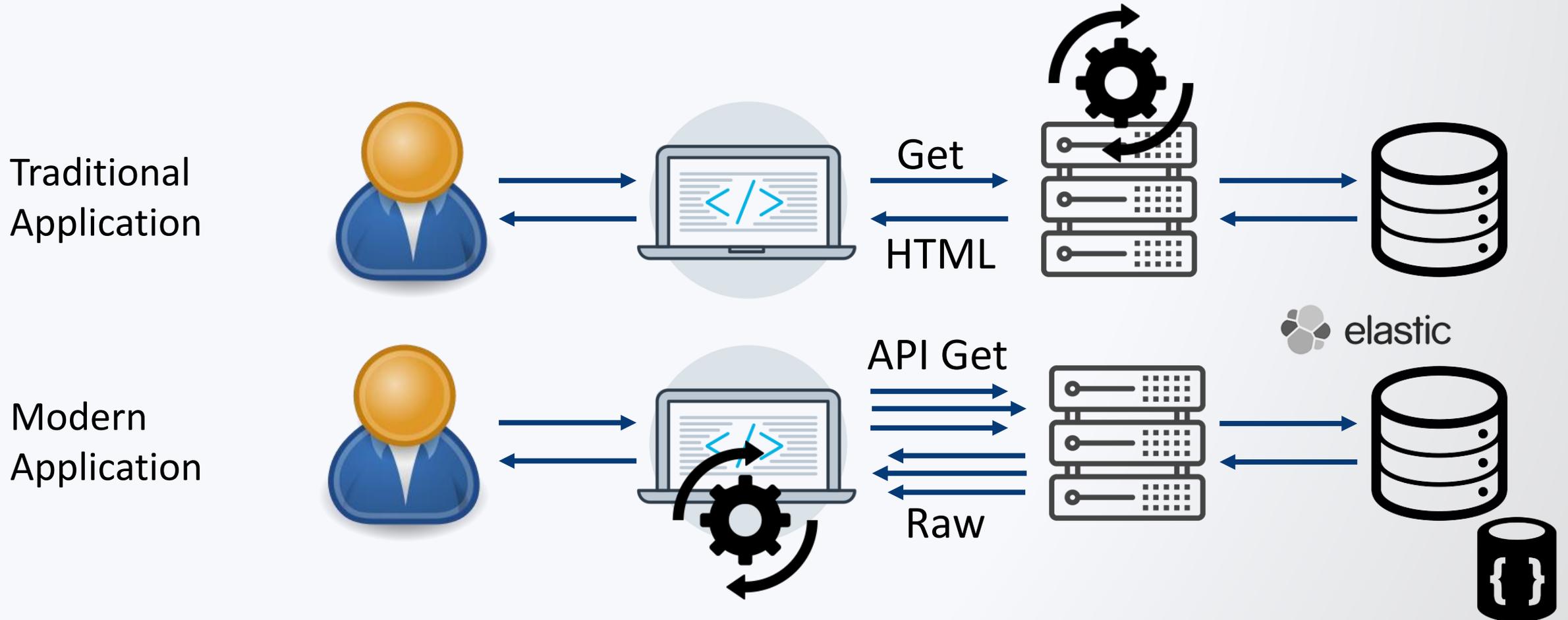
Traditional vs. Modern

Less abstraction layers

Client and server (and DB) speak the same JSON language

Modern Application

elastic

API Get

Raw

JSON

JSON

JSON
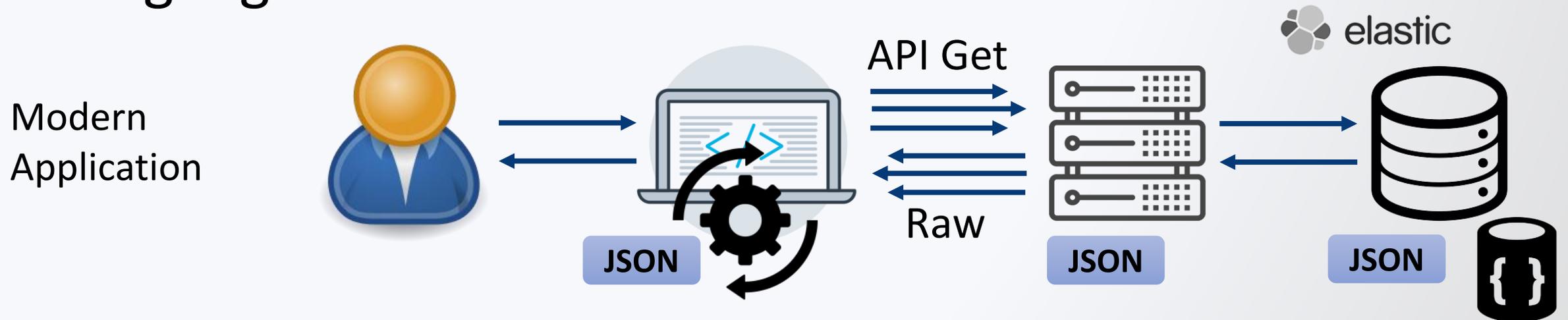
How API Based Apps are Different?

- The server is used more as a proxy for data
- The rendering component is the client, not the server

- Clients consume raw data
- APIs expose the underlying implementation of the app
- The user's state is usually maintained and monitored by the client
- More parameters are sent in each HTTP request (object ID's, filters)

How API Based Apps are Different?

- The REST API standard
  - Standardized & generic
  - Predictable entry points
  - One entry point (URL) can be used for multiple purposes
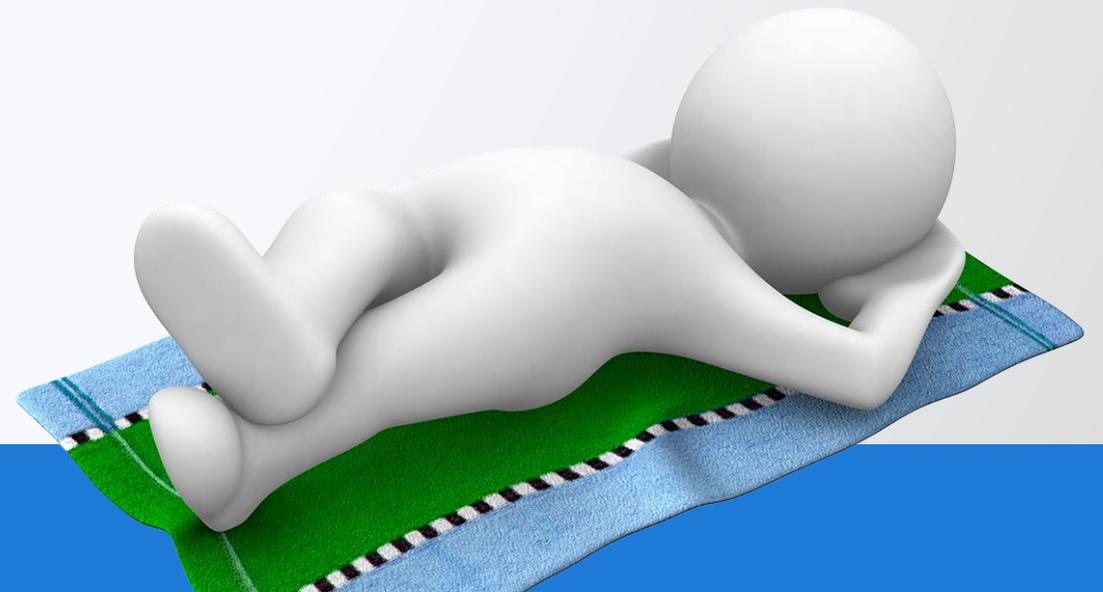
How API Based Apps are Different?

The good news

Traditional vulnerabilities are less common in API-Based apps:
- SQLi – Increasing use of ORMs
- CSRF – Authorization headers instead of cookies
- Path Manipulations – Cloud-Based storage
- Classic IT Security Issues - SaaS

What About Dev(Sec)Ops?

APIs change all the time
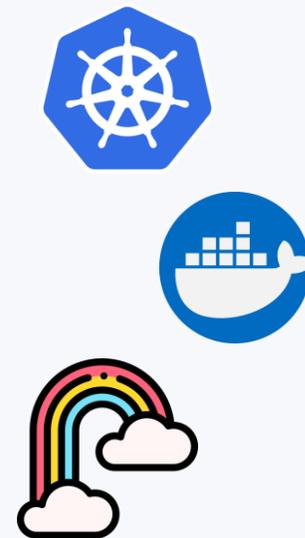
It takes just a few clicks to spin up new APIs (hosts). Too easy!

APIs become hard to track:

- Shadow APIs
- Old Exposed APIs

Roadmap – Planned Projects

- API Secrity Top 10
- API Security Cheat Sheet
- crAPI (**C**ompletely **R**idiculous **API**
  - an intentionally vulnerable API project)

Roadmap

	Top 10	Cheat Sheet	crAPI
2019 Q1	Prepare		
2019 Q2	Kick-Off		
2019 Q3	V1.0	Kick-Off	Prepare
2019 Q4		Collaborate	Kick-Off
2020 Q1		V1.0	Collaborate
2020 Q2			V1.0

The creation process of the Top10

- Internal knowledge and experience

- Internal data collection (Bug bounties reports, published incidents, etc.)

- Call for Data

- Call for comments

API Security Top 10

- **A1**: Broken Object Level Authorization
- **A2**: Broken Authentication
- **A3**: Excessive Data Exposure
- **A4**: Lack of Resources & Rate Limiting
- **A5**: Broken Function Level Authorization
- **A6**: Mass Assignment
- **A7**: Security Misconfiguration
- **A8**: Injection
- **A9**: Improper Assets Management
- **A10**: Insufficient Logging & Monitoring

Authz in APIs - The Challenge

- Decentralized Mechanism

Object Level	Function Level
Code (Almost every controller)	Code, Configuration, API-gateway

- Complex Users & Roles Hierarchies

Riders	Drivers	Admins
Hugo Sub #1 Sub #2 Bugo	Jon	Inon Shredder
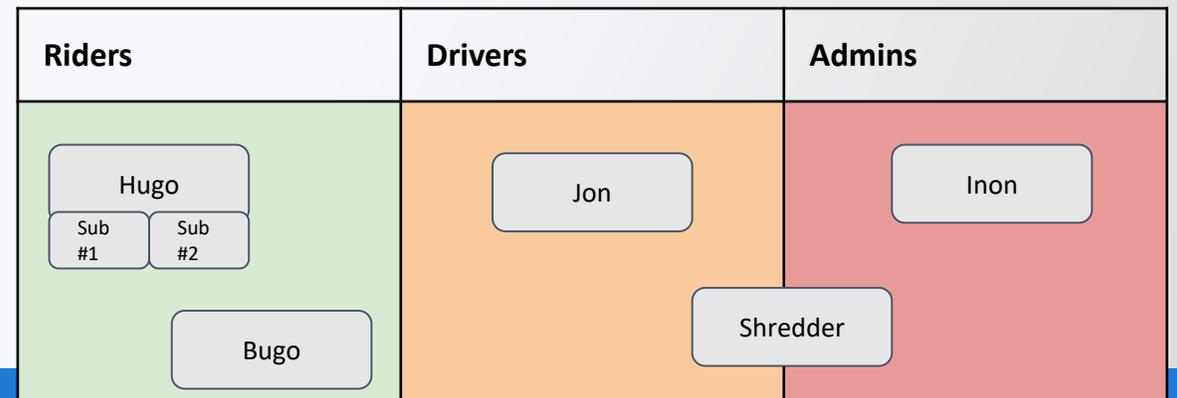
A1 - BOLA (Broken Object-Level Authorization)

Trip.find_by_id(718492).update

Rate your ride

★★★★★

GREAT

Awesome! What went well?

| Good Driving | Friendly Driver |
| Clean Car | Fun Conversation |

Share with NotLyft

Feedback is anonymous – we'll review it before sharing anything with your driver. Learn more
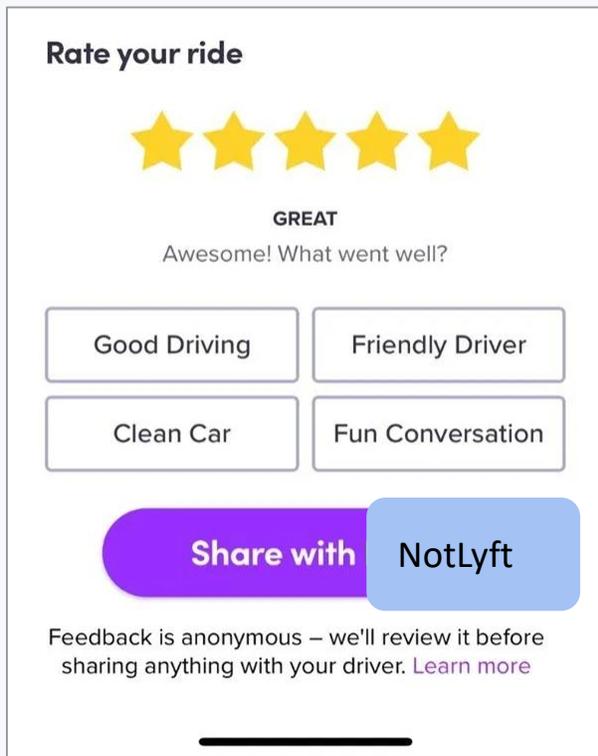
POST api/trips/rate_trip
{"**trip_id**":718492, "rate":5}

API
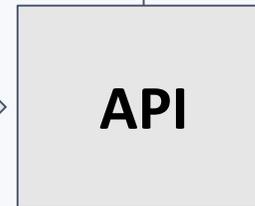
UPDATE trips …
WHERE ID = 718492

DB
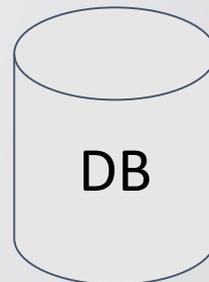
BOLA - Why Not IDOR

- **IDOR** - **I**nsecure **D**irect **O**bject **R**eference

- C00L name, not accurate

- The problem is not about the IDs !

BOLA - Solutions that <u>don't</u> solve the problem

- GUIDs instead of numbers
- Indirect Object Reference
- Relying on IDs from JWT tokens

BOLA - Solutions that <u>solve</u> the problem

- Good authorization mechanism
- Make sure that developers actually use it in every controller

BOLA - Uber - Full Account Takeover

Request

```
POST /marketplace/\_rpc?rpc=getConsentScreenDetails HTTP/1.1
Host: bonjour.uber.com
Connection: close
Content-Length: 67
Accept: application/json
Origin: [https://bonjour.uber.com](https://bonjour.uber.com)
x-csrf-token: xxxx
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleW
DNT: 1
Content-Type: application/json
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: xxxxx
{"language":"en","userUuid":"xxxx-776-4xxxx1bd-861a-837xxx604ce"}
```

Found by Anand Prakash,
AppSecure

Response

```
{
  "status":"success",
  "data":{
    "data":{
      "language":"en",
      "userUuid":"xxxxxx1e"
    },
    "getUser":{
      "uuid":"cxxxxxc5f7371e",
      "firstname":"Maxxxx",
      "lastname":"XXXX",
      "role":"PARTNER",
      "languageId":1,
      "countryId":77,
      "mobile":null,
      "mobileToken":1234,
      "mobileCountryId":77,
      "mobileCountryCode":"+91",
      "hasAmbiguousMobileCountry":false,
      "lastConfirmedMobileCountryId":77,
      "email":"xxxx@gmail.com",
      "emailToken":"xxxxxxxx",
```

A2 - Broken Authentication

forgot_password	EXTRA Protection	Rate Limiting (A4)
web_login		
get_location		
update_picture		

Lack of protection:
- Account lockout
- Captcha
- Brute Force attacks

Misconfiguration:
- JWT allows {"alg":"none"}
- Tokens don't expire
- etc..

A2 - Facebook - Full Account Takeover

V

Lockout mechanism
(after 10 attempts)

facebook.com

X

beta.facebook.com

X

mbasic.beta.facebook.com

Found by Anand Prakash,
AppSecure

Vulnerable request:

POST /recover/as/code/ HTTP/1.1
Host: beta.facebook.com

lsd=AVoywo13&n=XXXXX   *(5 Digits Reset Password Token)*
                              100,000 options

Brute forcing the "n" successfully allowed me to set new password for any Facebook user.

A3 - Excessive Data Exposure

- APIs expose sensitive data of other Users by design

A3 - Excessive Data Exposure

DATING APP

Name: Bob
Hobbies: Banana

GET /users/717

```
200 OK
{
    "name":"Bob",
    "hobbies":["Banana"],
    "profile_pic":"/bob.jpg",
    "address":"Gru's Mansion, 1000 Evil Rd"
}
```

X

API

Filtering sensitive information on the client side == **BAD IDEA!!**

A3 - Why ?

- API Economy + REST Standard == Generic Endpoints

- "to_ json" functions from ORM / Model

- Developers don't think who's the consumer

Recent Example - "3fun" app
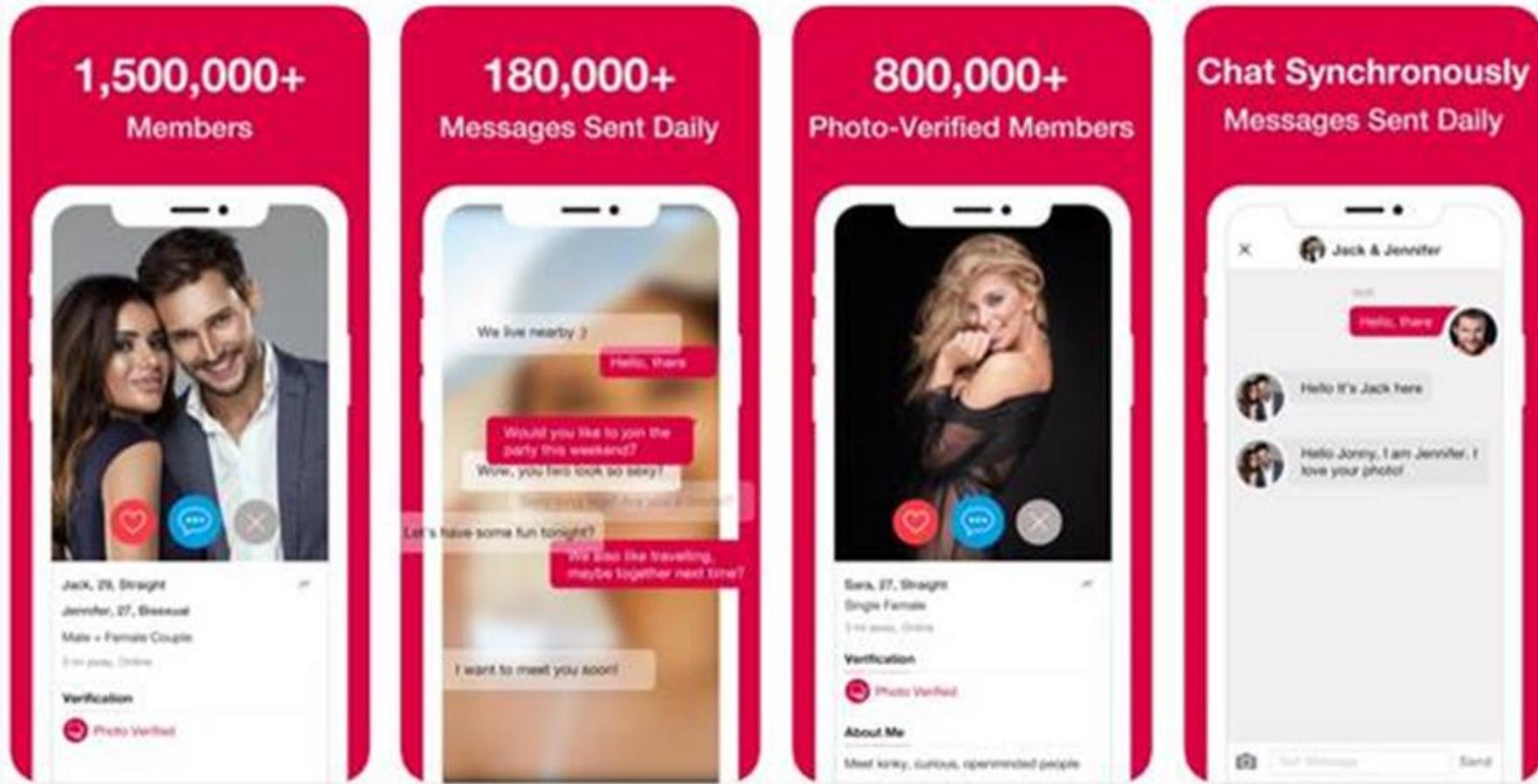
Found by Alex Lomas, Pen Test Partners

#	Host	Method	URL	Params	Edited	Status	Length	MIME type
322	https://www.go3fun.co	POST	/account_kit_reg	✓		200	447	JSON
325	https://www.go3fun.co	POST	/user/device_token	✓		200	198	JSON
326	https://www.go3fun.co	POST	/user/update	✓		200	265	JSON
327	https://www.go3fun.co	POST	/reset_push_badge			200	198	JSON
329	https://www.go3fun.co	GET	/match_users?from=0&latitude=51.▮▮▮...	✓		200	23807	JSON
331	https://www.go3fun.co	GET	/user/refresh			200	788	JSON
334	https://www.go3fun.co	POST	/user/update_location	✓		200	198	JSON
338	https://www.go3fun.co	POST	/upload_photo	✓		200	479	JSON
339	https://www.go3fun.co	GET	/i_like_list?from=0&offset=30	✓		200	201	JSON
340	https://www.go3fun.co	GET	/chatted_list			200	201	JSON
341	https://www.go3fun.co	POST	/reset_push_badge			200	198	JSON
344	https://www.go3fun.co	GET	/user/refresh			200	992	JSON
348	https://www.go3fun.co	GET	/matched_list?from=0&offset=30	✓		200	201	JSON

Request Response

Raw Headers Hex JSON Beautifier

        },
        "latitude": "51.▮▮▮▮",
        "membership": "2",
        "birthday": "1977-▮▮▮▮",
        "sex_orient": "4",
        "gender": "1",
        "longitude": "-0.1▮▮▮▮",
        "photo_verified_status": "1",
        "active": "0",
        "partner_sex_orient": "0",
        "liked_me": "0",
        "settings": {
          "show_online_status": "1",
          "show_distance": "1"
        },
        "username": "▮▮▮▮▮▮",
        "usr_id": "17▮▮▮▮",
        "about_me": "Kinky and attractive french financier open to many things ..."
      },
      {
        "last_login": "2019-06-24 20:21:12",
        "private_photos": [
          {
            "icon": "https://s3.amazonaws.com/3fun/821/▮▮▮▮▮▮/▮▮▮▮-small.jpg",
            "photo_id": "38▮▮▮▮",

Found by Alex Lomas,
Pen Test Partners

OWASP GLOBAL APPSEC
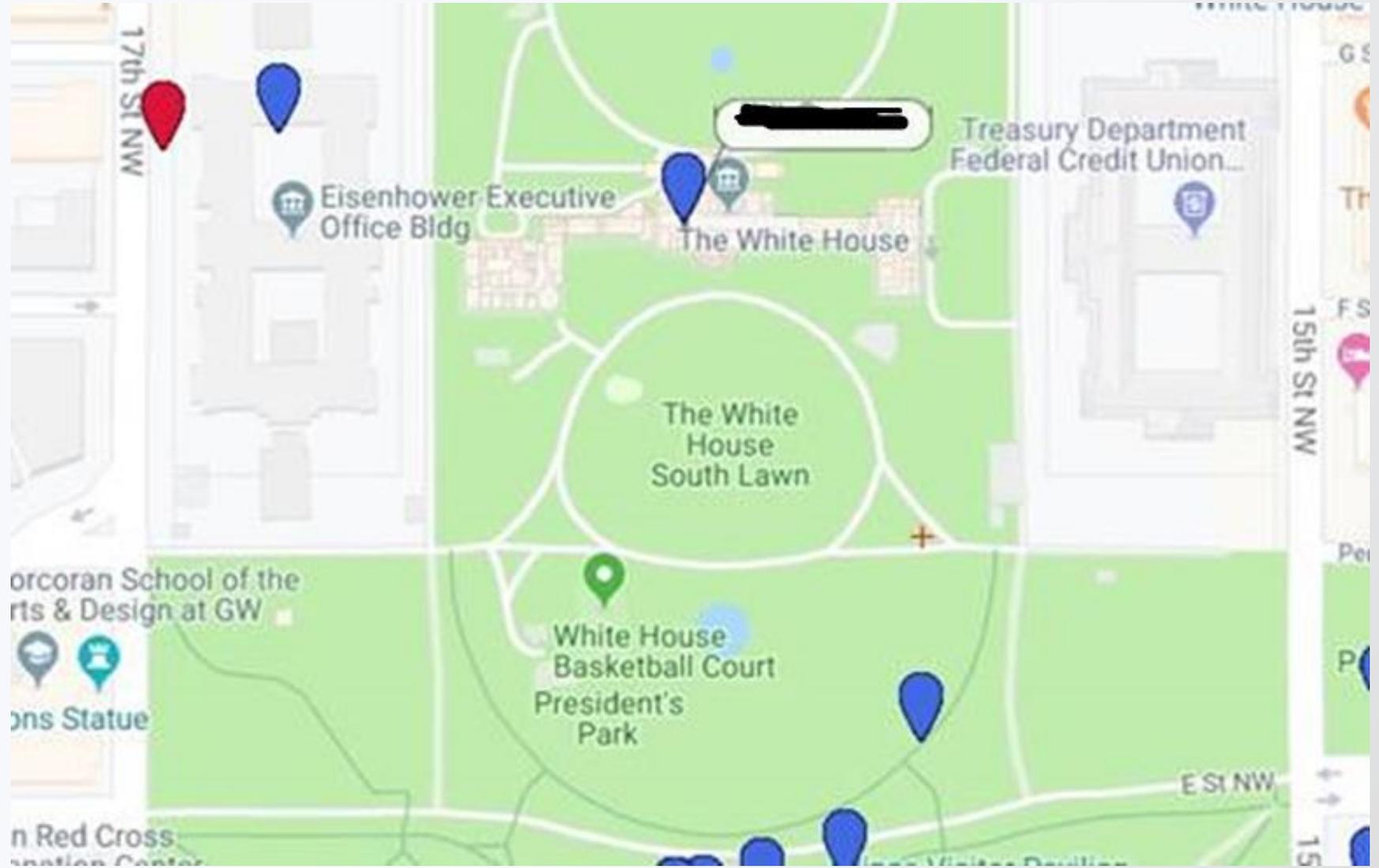
Found by Alex Lomas,
Pen Test Partners

A4 - Lack of Resources & Rate Limiting

- Might lead to DOS
- www.socialnetwork.com/users/list?limit=**99999999**

A5 - BFLA
(Broken Function Level Authorization)

Admin

DELETE /users/717

**Attacker
w/ Driver User**

DELETE /users/717

Admin API	
Riders API	**API**
Drivers API	
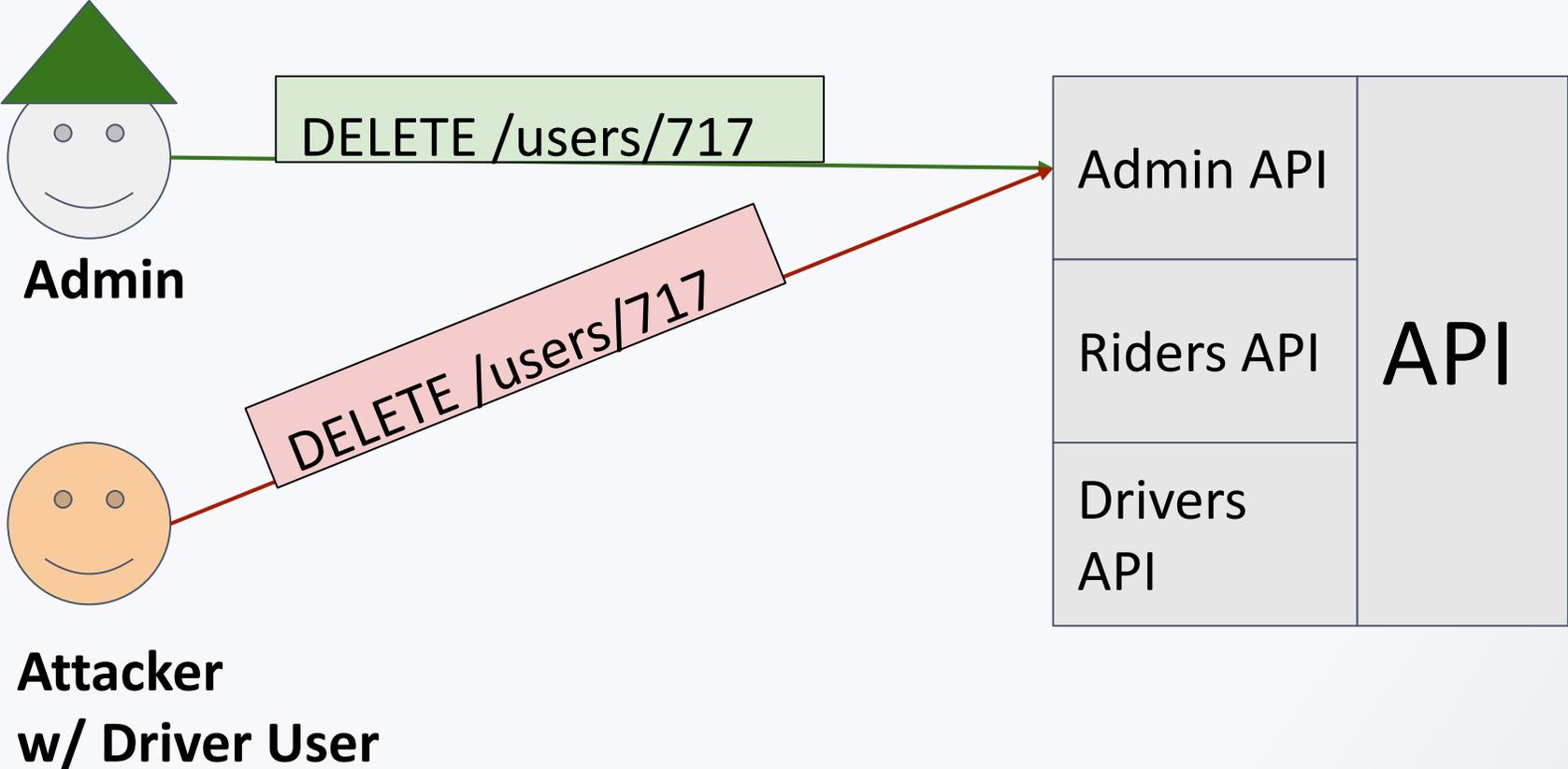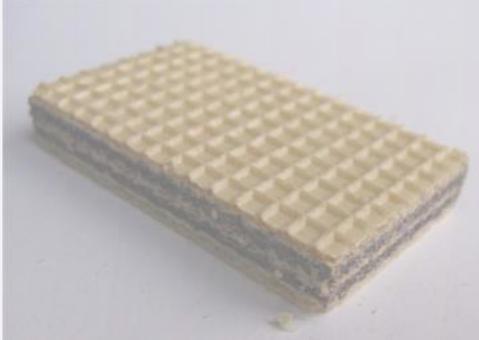
Why in APIs

	Fetch User's Profile (not sensitive function)	Delete user (admin function)
Traditional App	GET /app/users_view.aspx?user_id=1337	POST app/admin_panel/users_mgmt.aspx action=delete&user_id=1337
API	**GET** /api/users/1337	**DELETE** /api/users/1337

HARD to predict :(

Very Predictable

Function Level Authorization

- Can be implemented in different components:
  - Code
  - Configuration
  - API Gateway

- Different Roles:
- Admins / Super-admins / supervisors / riders / drivers

A5 - BFLA - Example - Shopify

@uzsunny reported that by creating two partner accounts sharing the same business email, it was possible to be granted "collaborator" access to any store without any merchant interaction.

"The code did not properly check
what type the existing account was"

Found by uzsunny
$20,000 bounty on Hackerone

A6 - Mass Assignment "Create_user" flow in traditional apps

```
User new_user = User();
new_user.first_name = Request.Query["fname"];
new_user.last_name = Request.Query["lname"];
new_user.pass = Request.Query["pass"];
new_user.Save();
```

ORM
{first_name=Inon
last_name=shkedy
pass=123456}

Create_user
fname=inon&
lname=shkedy&
pass=123456

APP
Server
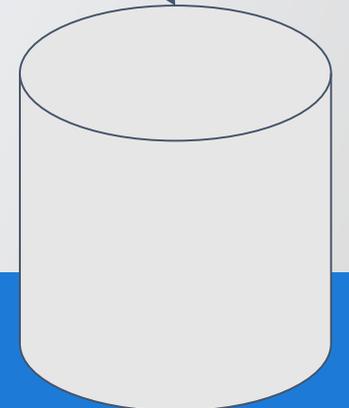
A6 - Mass Assignment

(ORM Black Magic)

`@user = User.new(params[:user])`

ORM
{JSON AS IS}

POST /users/create
{"user":{"lname":"Inon","fname":"shkedy","pass":"123456"}}

APP Server
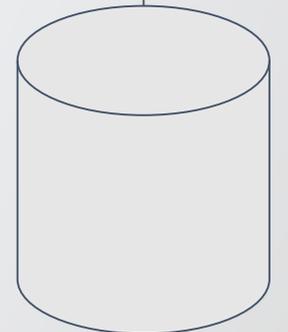
A6 - Mass Assignment

POST /api/users/new

{"username":"Inon", "pass":"123456"}

Legit

POST /api/users/new

{"username":"Inon", "pass":"123456", "role":"admin"}

Malicious

A6 - Why in APIs

- Mass Assignment isn't a new vulnerability.

- Easier to exploit in APIs though

- Don't guess object properties, just find a GET method that returns them :)

```
GET /v1/user/video_files
---------------------------
200 OK
{
    "id": 371,
    "name": "clip.mp4",
    "conversion_params":"-v codec h264"
}
```

```
PUT /v1/videos/371
{
    "name": "clip.mp4",
    "conversion_params":"-v codec h264 && format C:/"
}
```
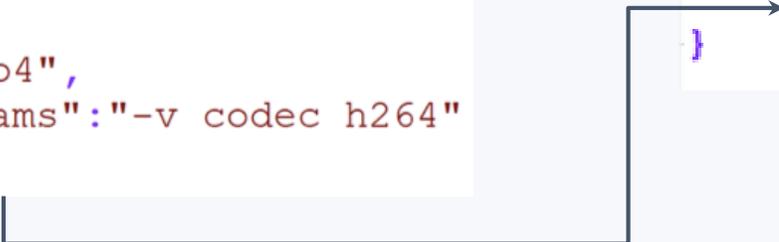
A6 - Example

James Kettle (albinowax)

2004 -
Reputation Rank

23 #267781 Users can enable API access for free via mass assignment

State   ● Resolved (Closed)   Severity
Disclosed July 9, 2019 1:08am +0200   Participants
Reported To New Relic   Visibility D
Weakness Privilege Escalation

```
POST /accounts/<account_id>.json

account[first_name]="Evil"&
account[allow_api_access]=true
```

New Relic

Found by
James Kettle,
Port Swigger

A7 - Security Misconfiguration

- Lack of CSRF / CORS protection

- Lack of security related HTTP headers

- Unnecessary exposed HTTP methods

- Weak encryption

- Etc...

Bad Things

A8 - Injection
Why from A1 to A8 ?

- First of all, ask yourself - **why injection was A1** ?

- SQLi much less common:
    - ORMs
    - Gazillion of security products that solve them
    - Use of NoSQL

- NoSQL Injection are a thing, but are usually not as severe / common

A9 - Improper Asset Management

API endpoints with no documentation

Unknown API hosts

API		
/v1/get_user	/v2/update_location	/v0/b2b_old/export_all_users

X
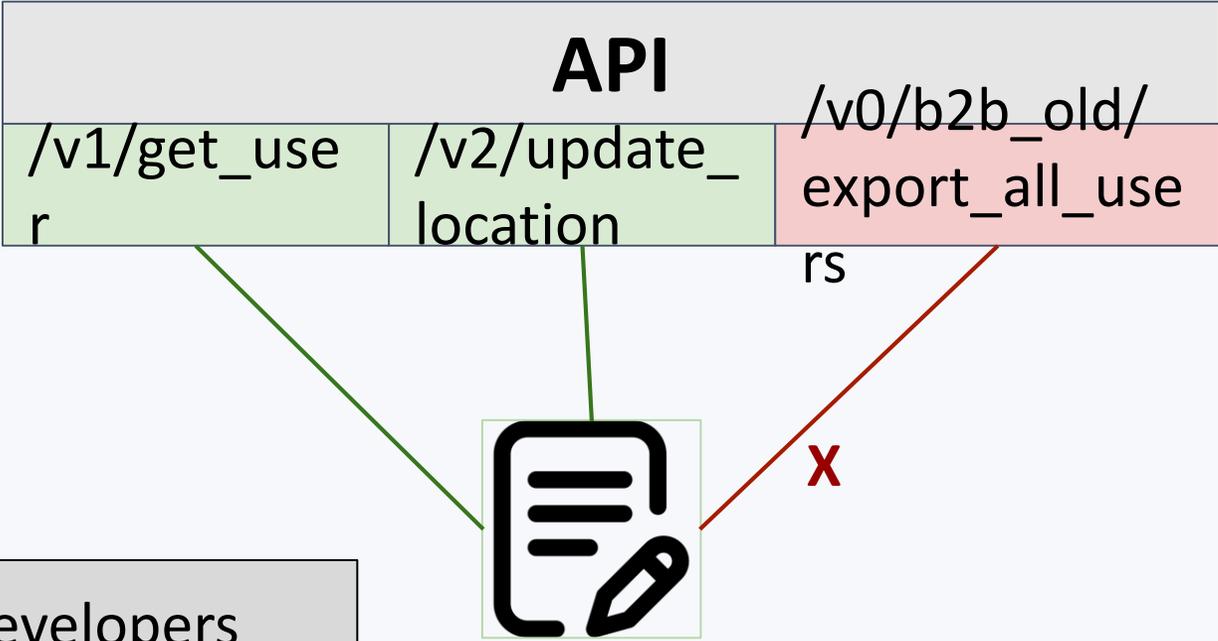
Developers

payment-api.acme.com

mobile-api.acme.com

qa-3-old.acme.com

DevOps

A9 - Why in APIs?

- APIs change all the time because of **CI/CD**

- Cloud + deployment automation (K8S) ==
Too easy to spin up a new API host

A10 - Insufficient Logging & Monitoring

- Same as A10 (2017)

Call for Discussions

Mailing List

https://groups.google.com/a/owasp.org/d/forum/api-security-project

Call for Contributions

GitHub Project

https://github.com/OWASP/API-Security/blob/develop/CONTRIBUTING.md

https://www.owasp.org/index.php/OWASP_API_Security_Project

https://github.com/OWASP/API-Security

QUESTIONS?