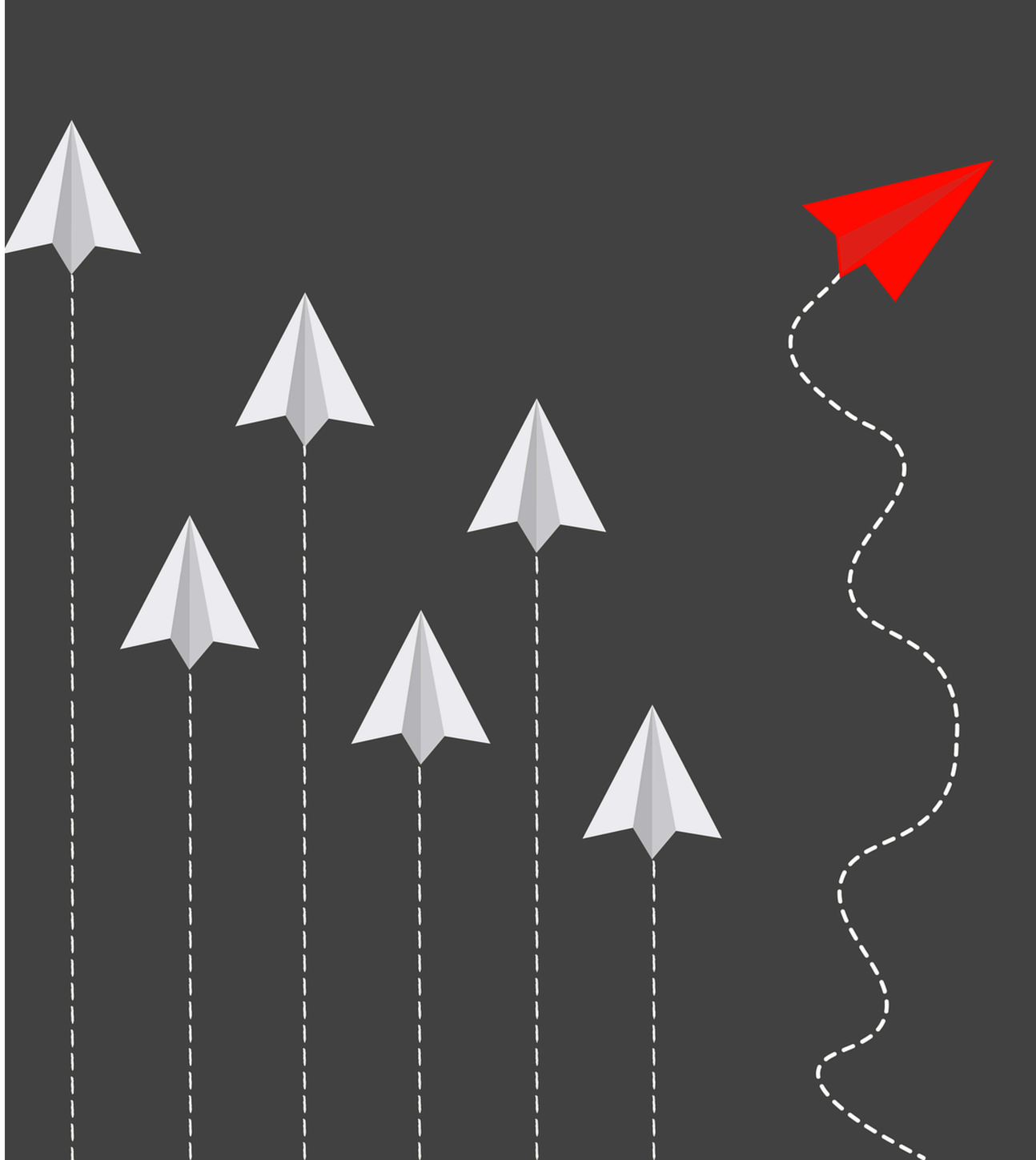

# Security Architecture in a Distributed World

ISABELLEMAUNY

Field CTO - 42Crunch

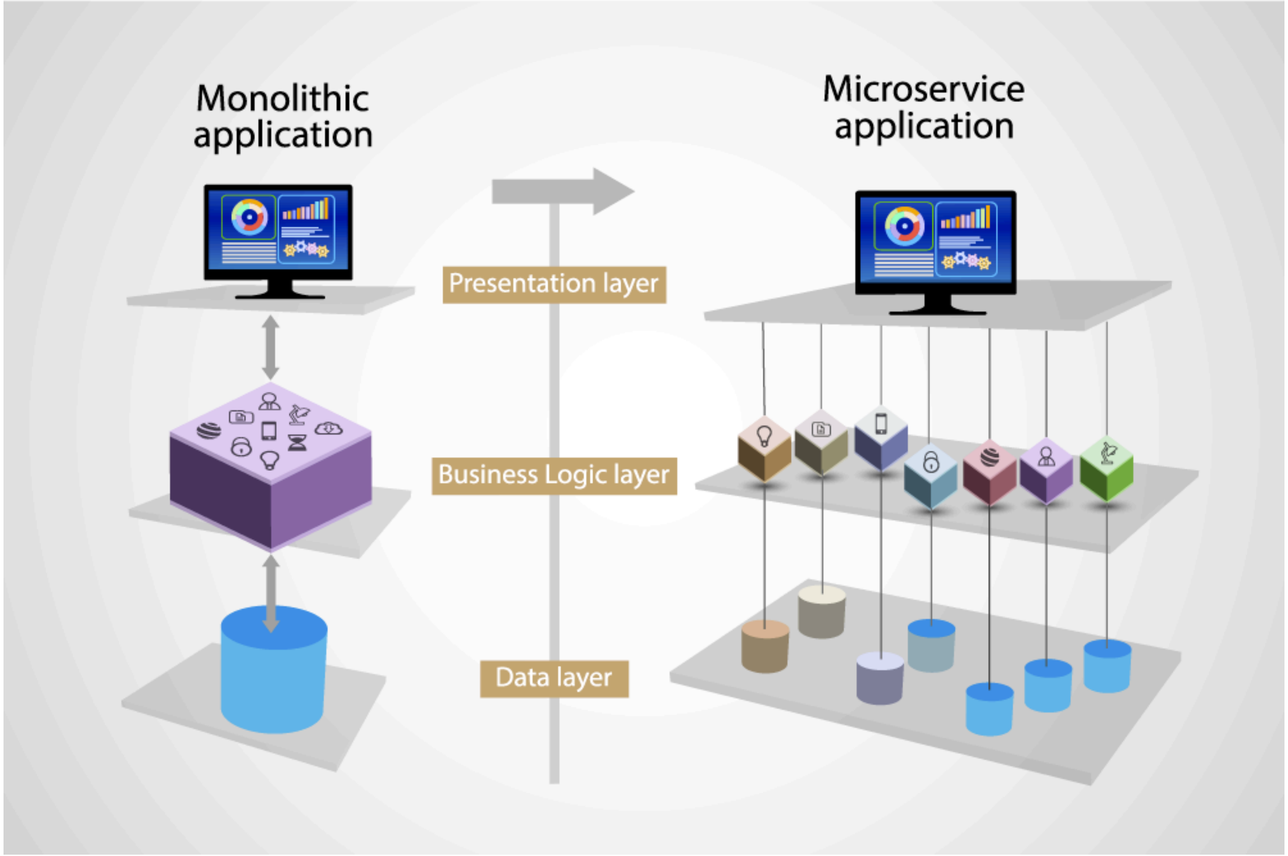
# Interactive Design Ahead

Be ready to participate!



# Fictitious Design Co

From our (not so) loved monolith to the moon !




Some guiding principles first!

An API **must not blindly trust** anything it receives or uses.

## And that includes

- Request payloads
- Headers
- JSON Web Tokens
- IP addresses (X-Forwarded-For)
- Libraries
- Docker Images
- 3rd party APIs
- Anything, really…

# OWASP API Security Top 10 2023

- API1:2023 Broken Object Level Access Control (aka *BOLA*)
- API2:2023 Broken Authentication
- API3:2023 Broken Object Property Level Authorization (**Updated**) (aka *BOPLA*)
- API4:2023 Unrestricted Resources Consumption
- API5:2023 Broken Function Level Authorization (aka *BFLA*)
- API6:2023 Server Side Request Forgery (**New**)
- API7:2023 Security Misconfiguration
- API8:2023 Lack of Protection from Automated Threats (**New**)
- API9:2023 Improper Assets Management
- API10:2023 Unsafe Consumption of APIs (**New**)

BAD PROBLEMS USUALLY OCCUR WHEN MULTIPLE OF THESE ARE COMBINED

# Authorization Levels

```
"/user/{userid}": {
  "put": {
    Scan | Try it
    "tags": [
      "users"
    ],
    "summary": "edit user information",
    "description": "user supplies valid token and updates all user info",
    "operationId": "edituserinfo",
    "parameters": [⋯
    ],
    "requestBody": {
      "description": "userobject",
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/UserUpdateData"
          }
        }
      },
      "required": true
    },
```

API 5 : BFLA

API 1: BOLA

API 3: BOPLA

# Parler Social Network (January 2021)

https://apisecurity.io/issue-116-facebook-parler-api-vulnerabilities-clairvoyance/

- The Attack
  - Not sure we can call this an attack, more like “write a loop and get the data”
- The Breach
  - 70 **TB** of user’s data
- Core Issues
  - No Authentication to access public posts
  - No Rate limiting
  - Sequential IDs
  - Leaked raw data about posts, including location
  - Deleted data was not deleted, just hidden in the UI

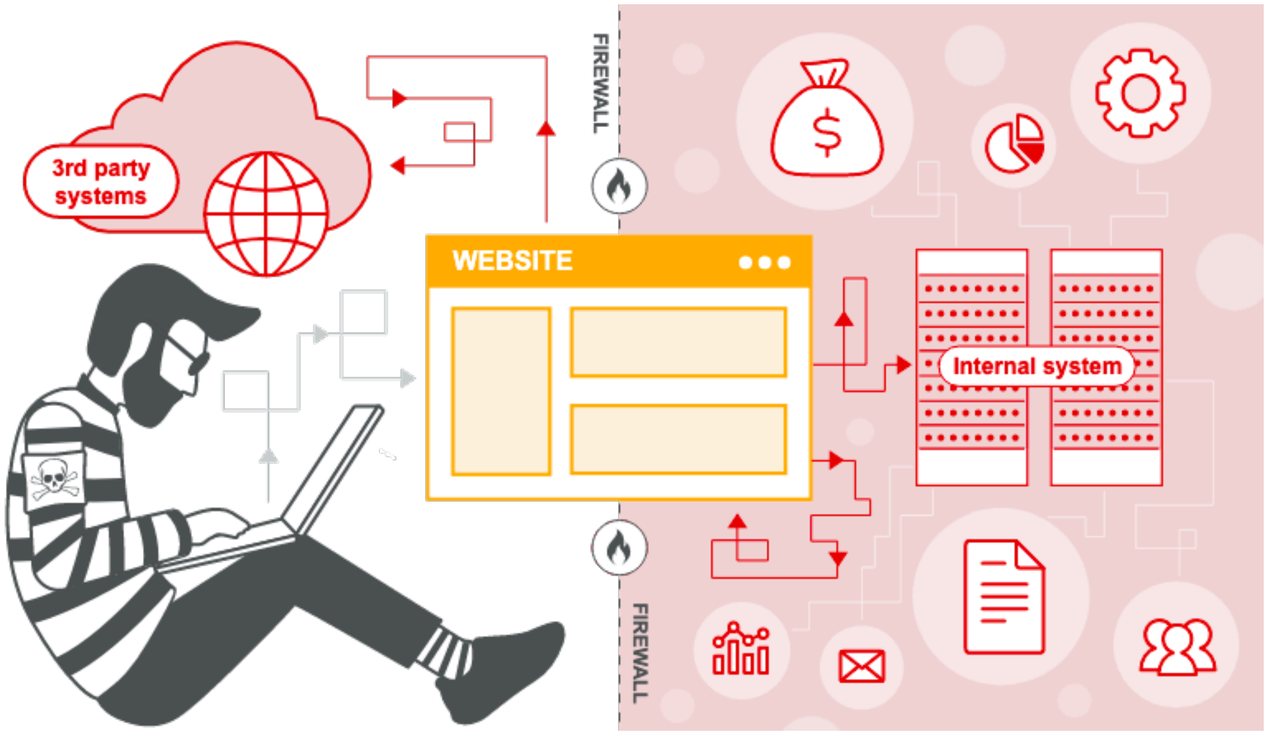
API1
API2
API3
API4
API5
API6
API7
API8
API9
API10

# New kid on the block: SSRF

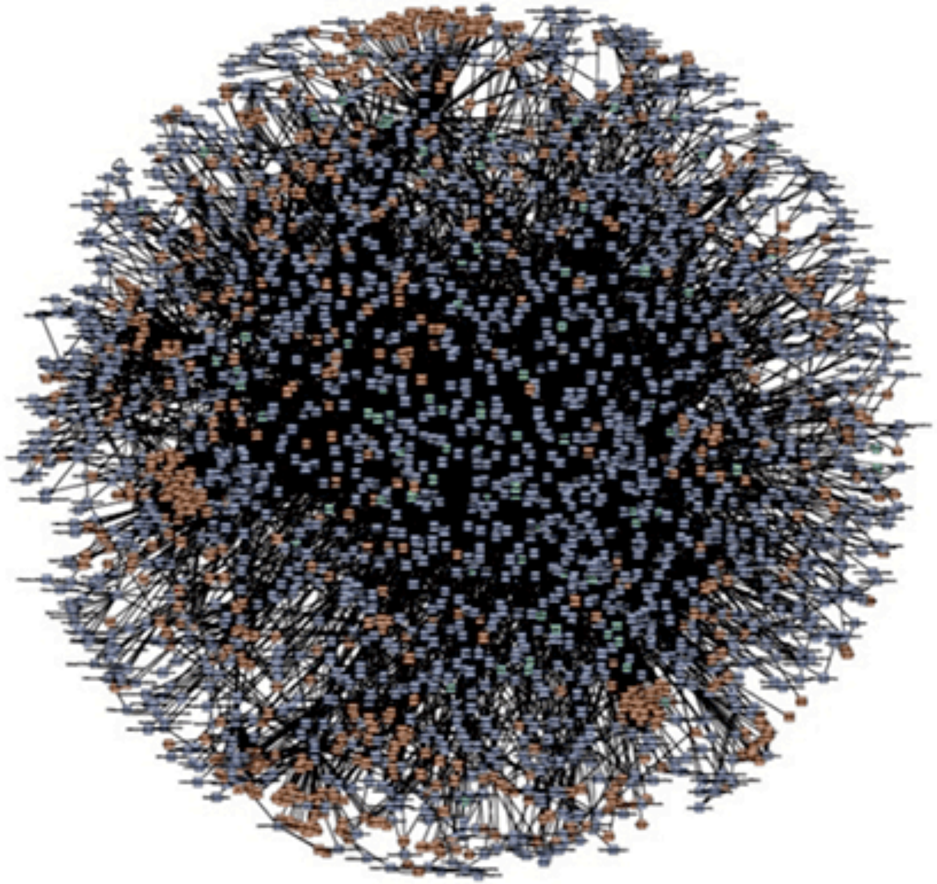
Server-side request forgery is a web security vulnerability that allows an attacker to **induce the server-side application to make requests to an unintended location.**

Great intro: https://danaepp.com/exploiting-ssrf-in-an-api

Recent example: https://ermetic.com/blog/azure/when-good-apis-go-bad-uncovering-3-azure-api-management-vulnerabilities/




So shall we use microservices ?!



amazon.com

NETFLIX

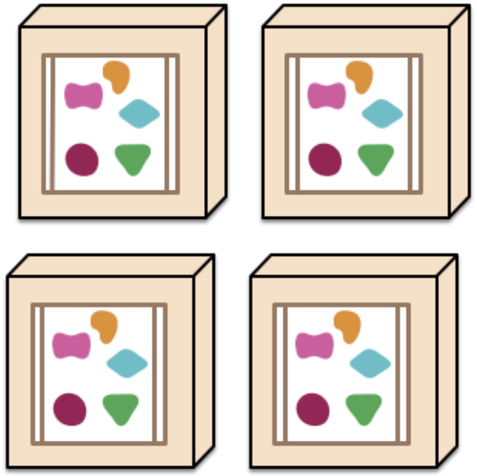
Will you build your own Death star ?

# Reasons to **use** micro/ macro/mini services

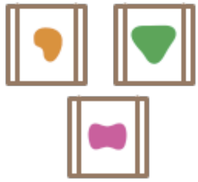
- Independent development of microservices
- Independent scaling of microservices
- Choice of development framework, even within the same application
- Cloud ready!

*A monolithic application puts all its functionality into a single process...*

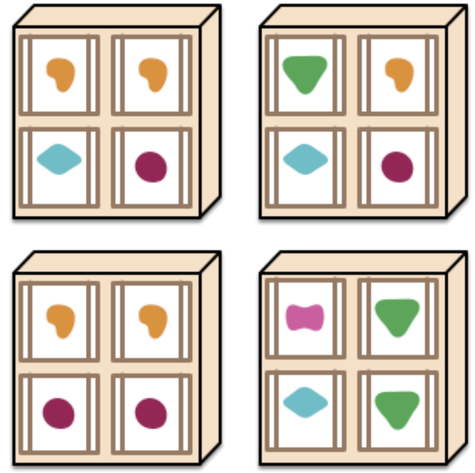
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*



*... and scales by distributing these services across servers, replicating as needed.*



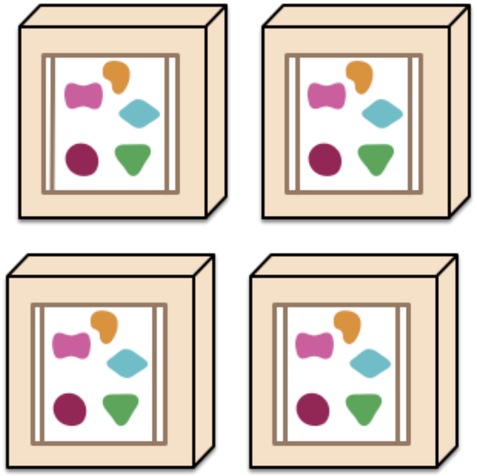
# Reasons to **not use** micro/ macro/mini services?

- Independent development of microservices
- Independent scaling of microservices
- Choice of development framework, even within the same application
- Not Cloud ready…

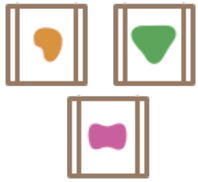
- **None of this is coming for free…**
- **In particular, your attack surface is going to explode!**

*A monolithic application puts all its functionality into a single process...*

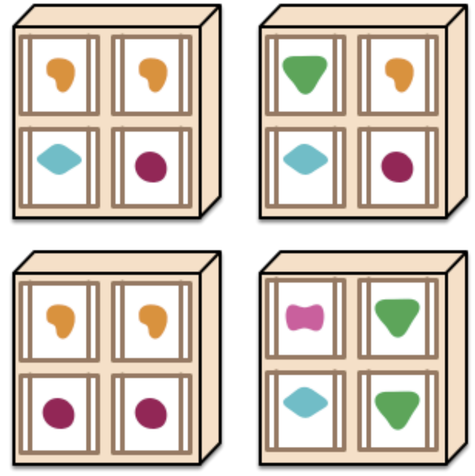
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*



*... and scales by distributing these services across servers, replicating as needed.*

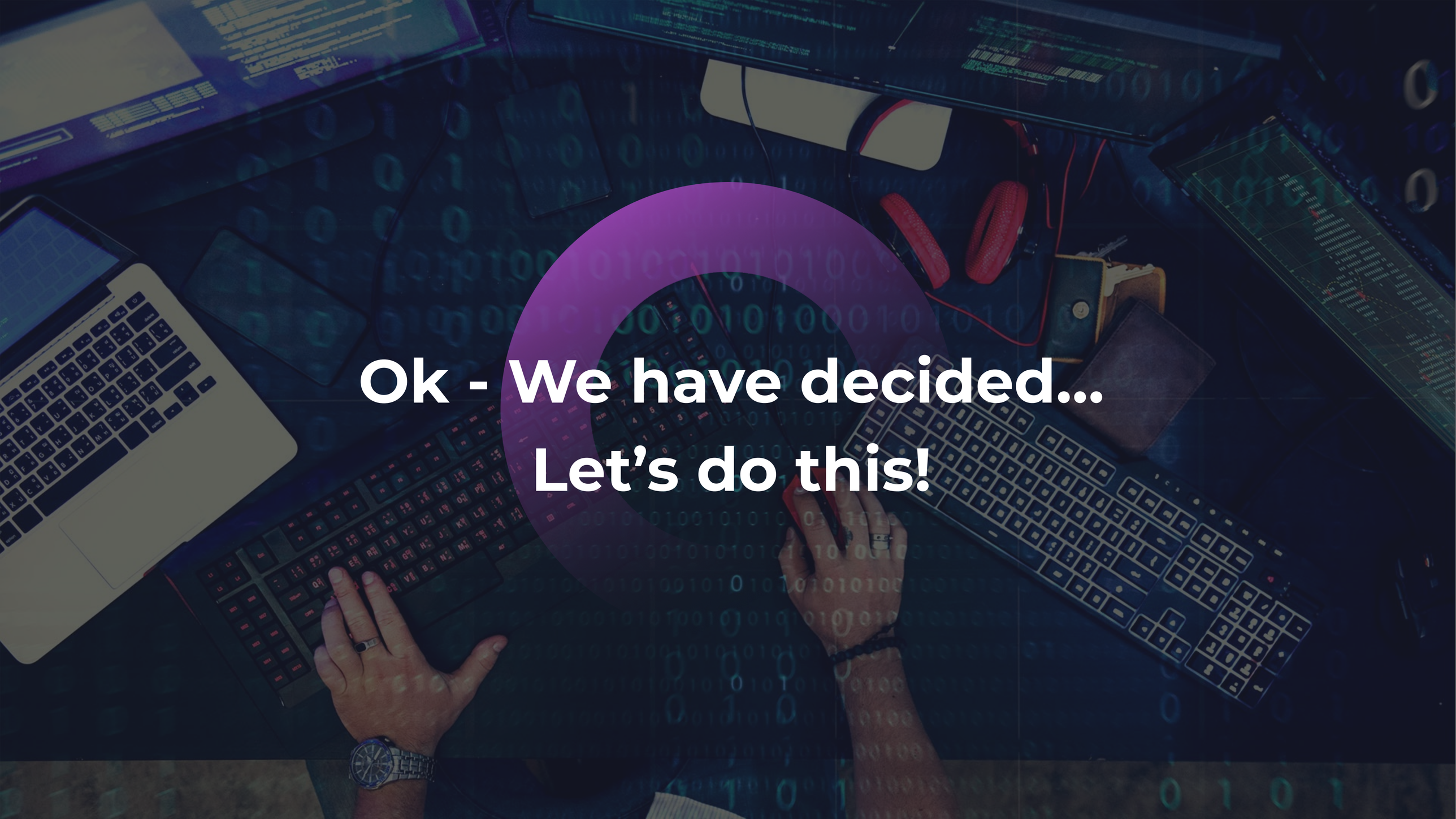


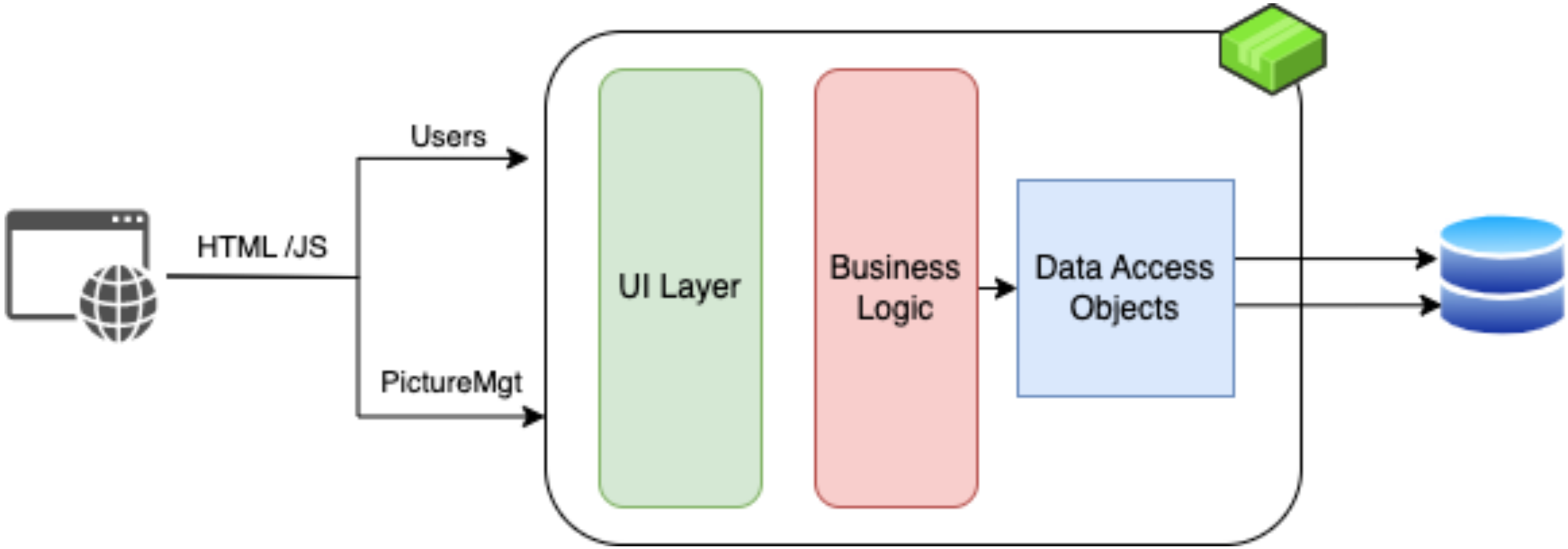
# What could go wrong ?

- **API 4**: Microservices talk over network, not over process calls - So now you need to manage:
  - Timeouts
  - Availability
- **API 4 / API 3**: Data layer exposed as a service - We need defense in depth
- **API 7**: Multiples languages/frameworks being used: need to master the supply chain across all of these plus the infrastructure.

# Operations Challenges

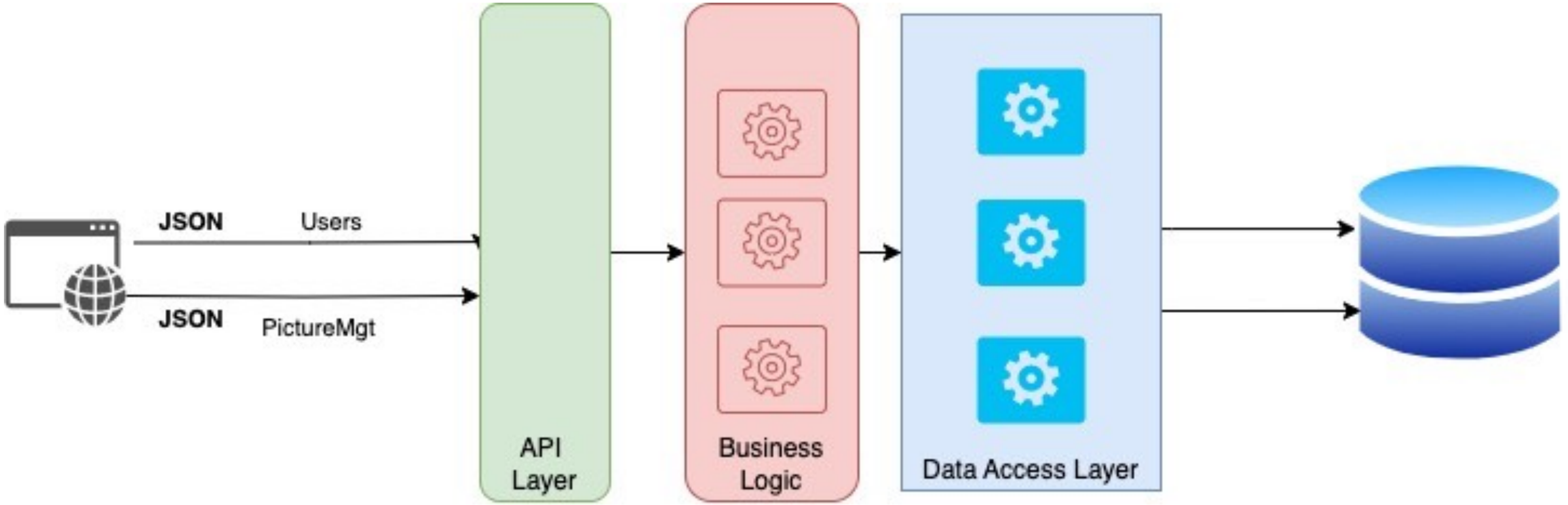
Automating (and securing) dozens of deployment per day.


Ok - We have decided...
Let’s do this!



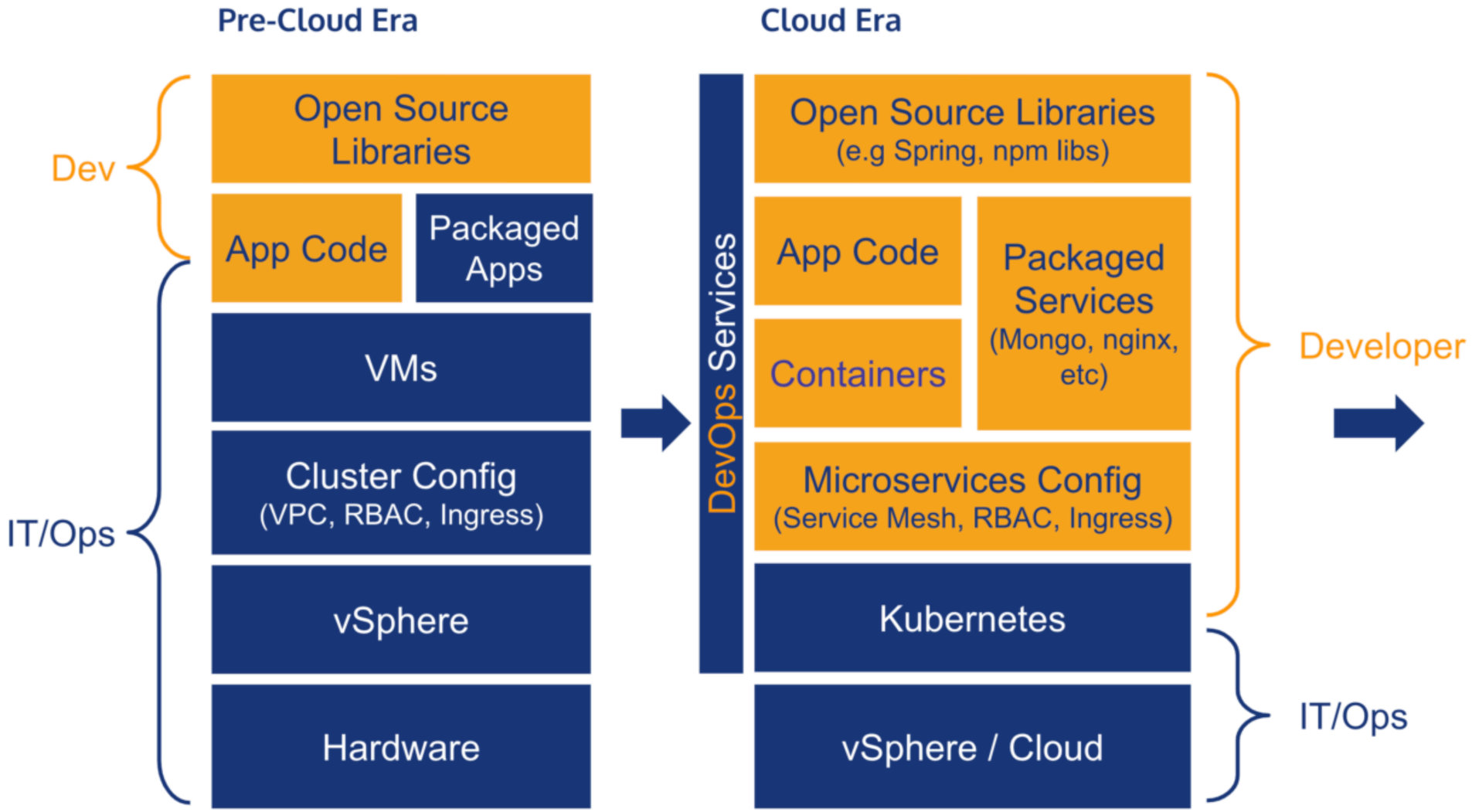

# Our scenario

Picture Sharing Application




# Target Architecture

Picture Sharing Application

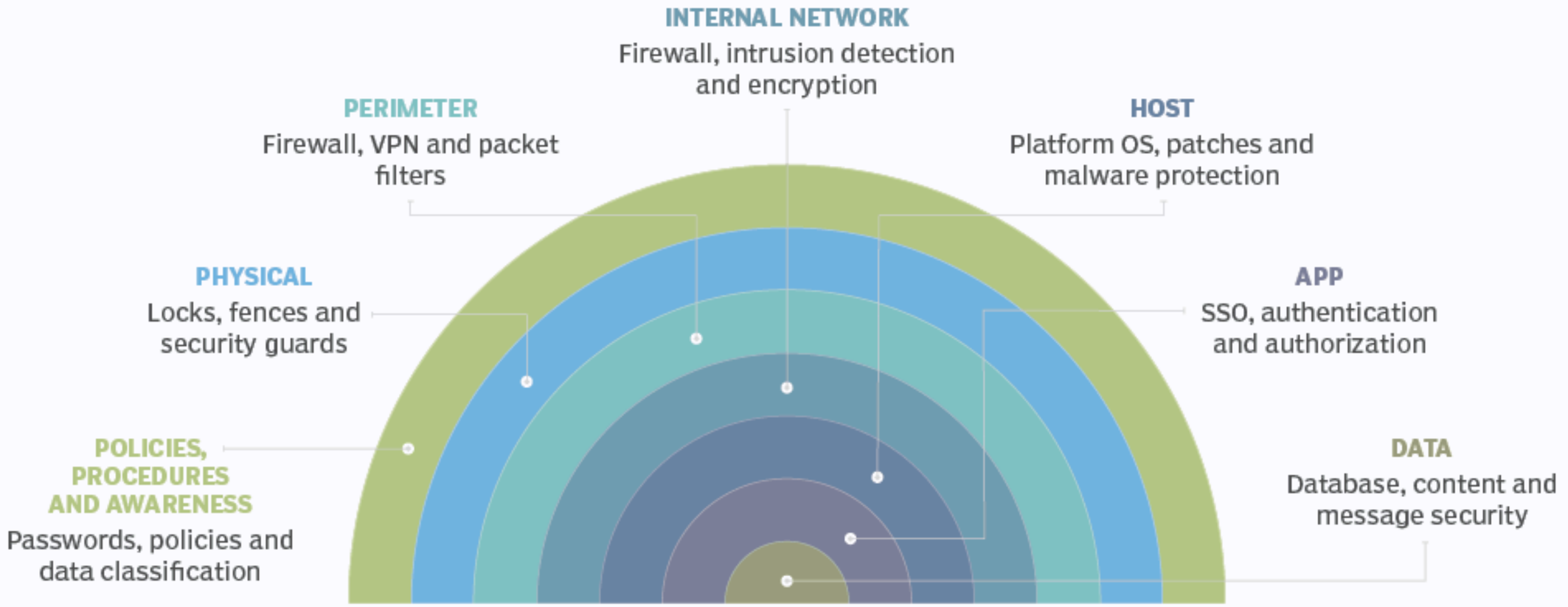



# The AppSec stack

Increased role/responsibility of developers.

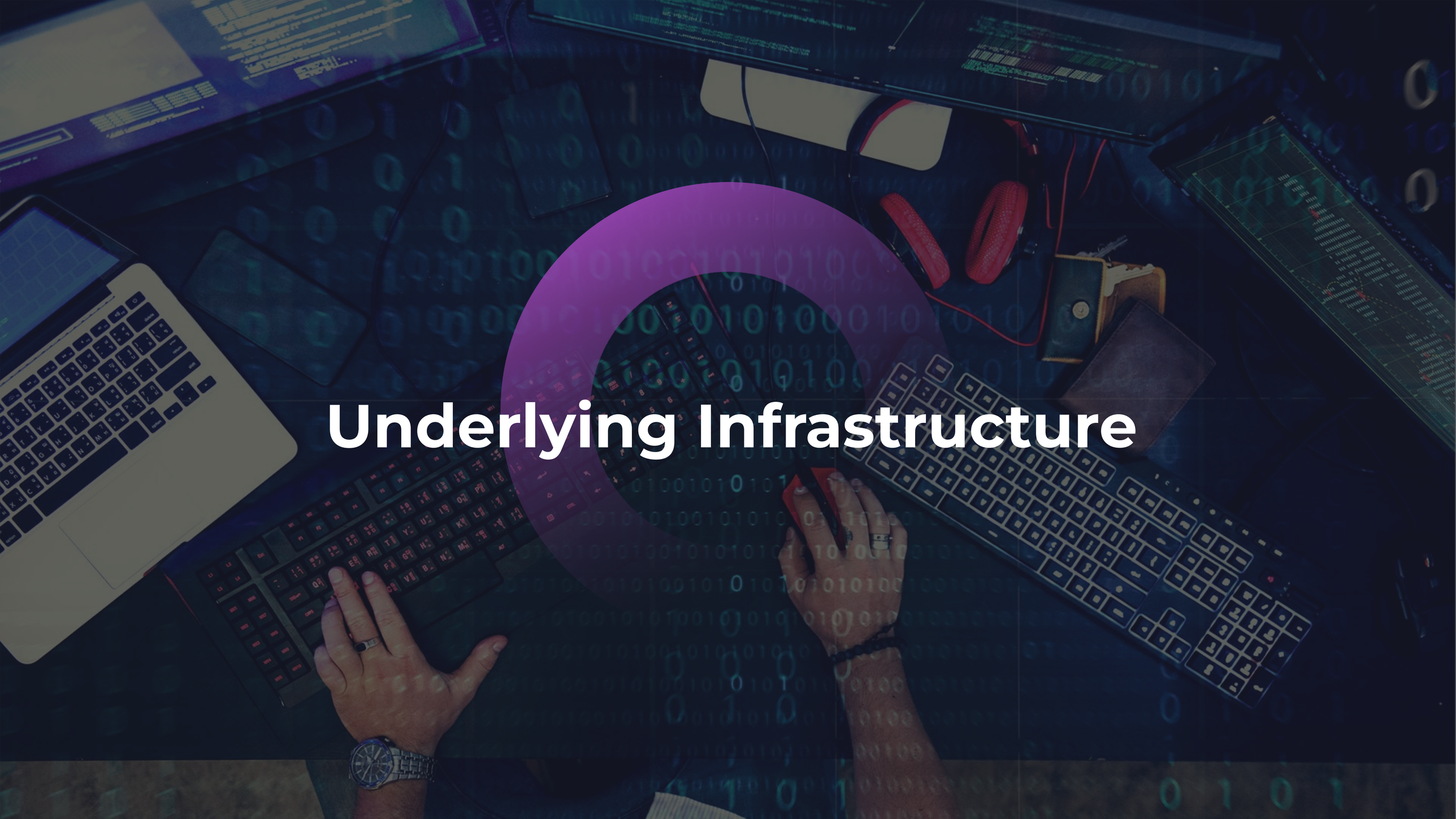
From: https://snyk.io/blog/cloud-transforms-it-security-appsec/

# Defense-in-depth layers



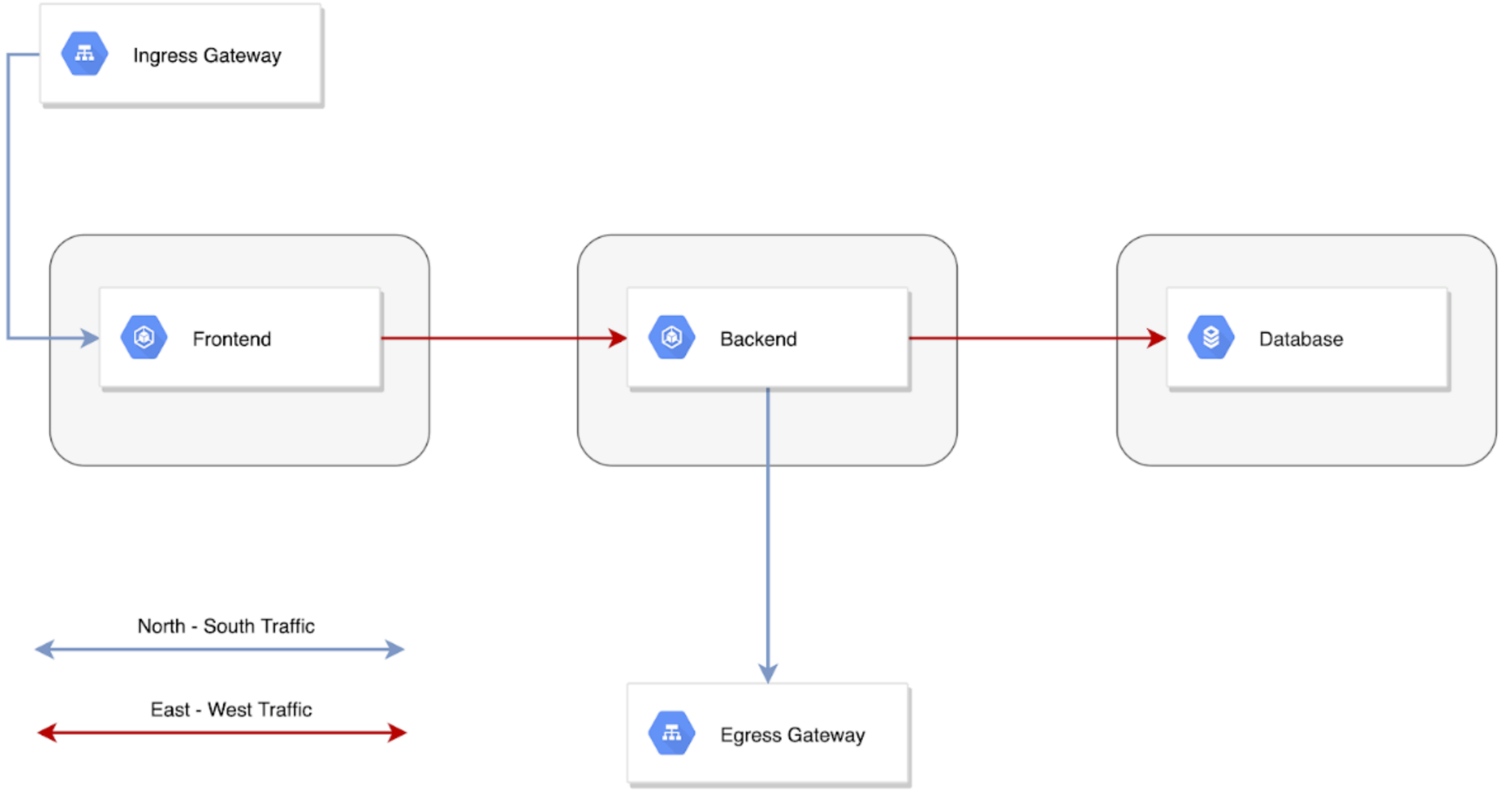

https://www.techtarget.com/searchsecurity/answer/Whats-the-difference-between-zero-trust-vs-defense-in-depth

Something  like this…



# Underlying Infrastructure

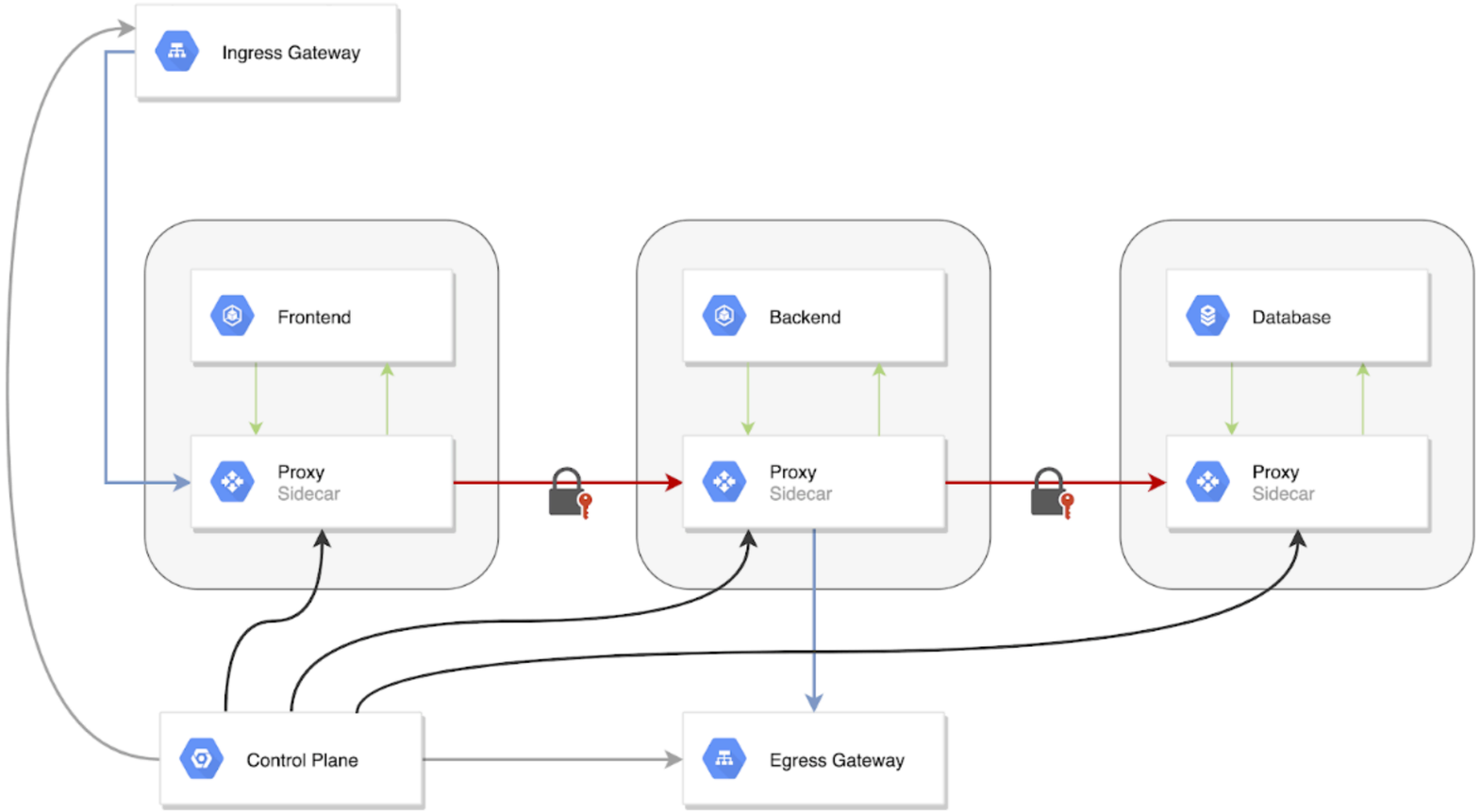
# Traffic overview




## Why a service mesh?

Traffic Routing / Encryption
Scalability
Resiliency
Observability
Service Discovery

# Service Mesh Traffic overview




## Implementing a service mesh

Traffic is intercepted and enforced according to centrally managed policies

https://www.weave.works/blog/introduction-to-service-meshes-on-kubernetes-and-progressive-delivery


Supply Chain Security

Closed · 5 tasks done · **alex-drocks** opened this issue on Oct 22, 2021 · 6 comments

**alex-drocks** commented on Oct 22, 2021 • edited

## 🐛 Bug Report

ua-parser-js version 0.7.29 and higher contain malware
faisalman/ua-parser-js#536 (comment)

### Prerequisites

- [x] I'm using the latest version of Docusaurus.
- [x] I have tried the `npm run clear` or `yarn clear` command.
- [x] I have tried `rm -rf node_modules yarn.lock package-lock.json` and re-installing packages.
- [x] I have tried creating a repro with https://new.docusaurus.io
- [x] I have read the console error message carefully (if applicable)

### Description

one of the dependency installed with npm install of the latest docusaurus version was hijacked by a malware executable file.
See above mentionned github issue link where you will get more details.

**Assignees**
No one assigned

**Labels**
bug

**Projects**
None yet

**Milestone**
No milestone

**Development**
No branches or pull requests

**Notifications** Customize

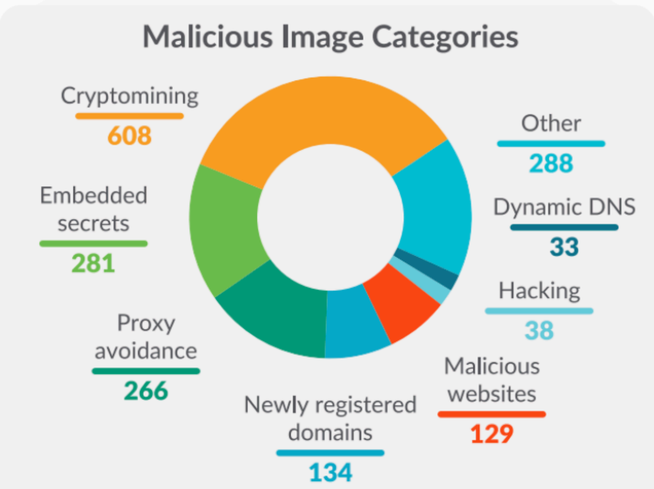
# UA Parser Hijacked

1600 NPM packages depend on this.
https://github.com/ua-parser/uap-core/blob/master/regexes.yaml

# Analysis on Docker Hub malicious images: Attacks through public container images

BY STEFANO CHIERICI - NOVEMBER 23, 2022

SHARE:




CONTENT: DOCKER HUB | TYPOSQUATTING, CRYPTOMINERS, AND KEYS | FINAL WORDS HIDE –

Supply Chain attacks are not new, but this past year they received much more attention due to high profile vulnerabilities in popular dependencies. Generally, the focus has been on the dependency attack vector. This is when source code of a dependency or product is modified by a malicious actor in order to compromise anyone who uses it in their own

Also true of Docker images


Mike Pittenger
Vice President, Security Strategy, Black Duck

Last week, the consumer reporting agency Equifax disclosed a major cybersecurity incident potentially affecting approximately 143 million US consumers.

The breach should never have happened.

Equifax acknowledged on Wednesday, Sept. 13, that a patch for the Apache Struts CVE-2017-5638 vulnerability—the culprit—was available in March, well before the attacks began. However, Equifax had not updated the vulnerable software at the time of the breach, more than two months later.

As a result, criminals were able to exploit the vulnerability and gain access to Equifax files from mid-May through July of this year, more than four months after the vulnerability had been disclosed publicly.



## More on Information Security

Stronger together

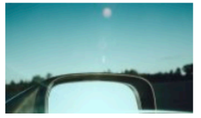
**RSA Conference 2023: Unity + Basics = Security**

by Rik Ferguson

Up close and personal

**The Case for ISO 27701-Compliant PII Management**

by Mitesh Karamchandani, Vishwameet Chawla



Everything everywhere

**The Blind Spots of Data-Regulation Compliance**

# Authentication/Authorization

# BrewDog Brewery

Full Vulnerability Report

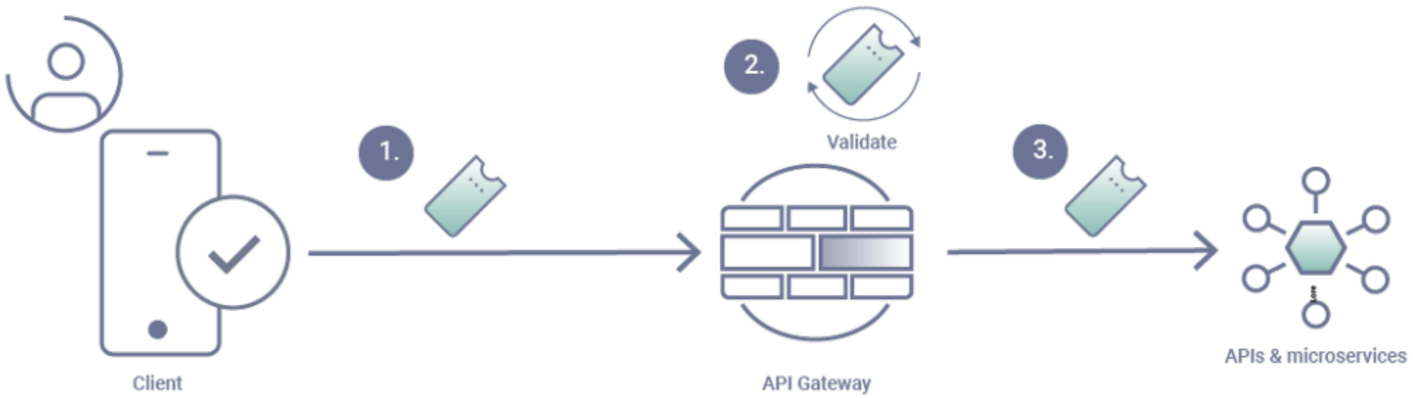
- The Attack
  - Complete take over.
- The Breach
  - Hard to tell…. Been like this for 18months+
  - Free beers!!
- Core Issues
  - Authentication token hardcoded in application. All calls come with the **same token**

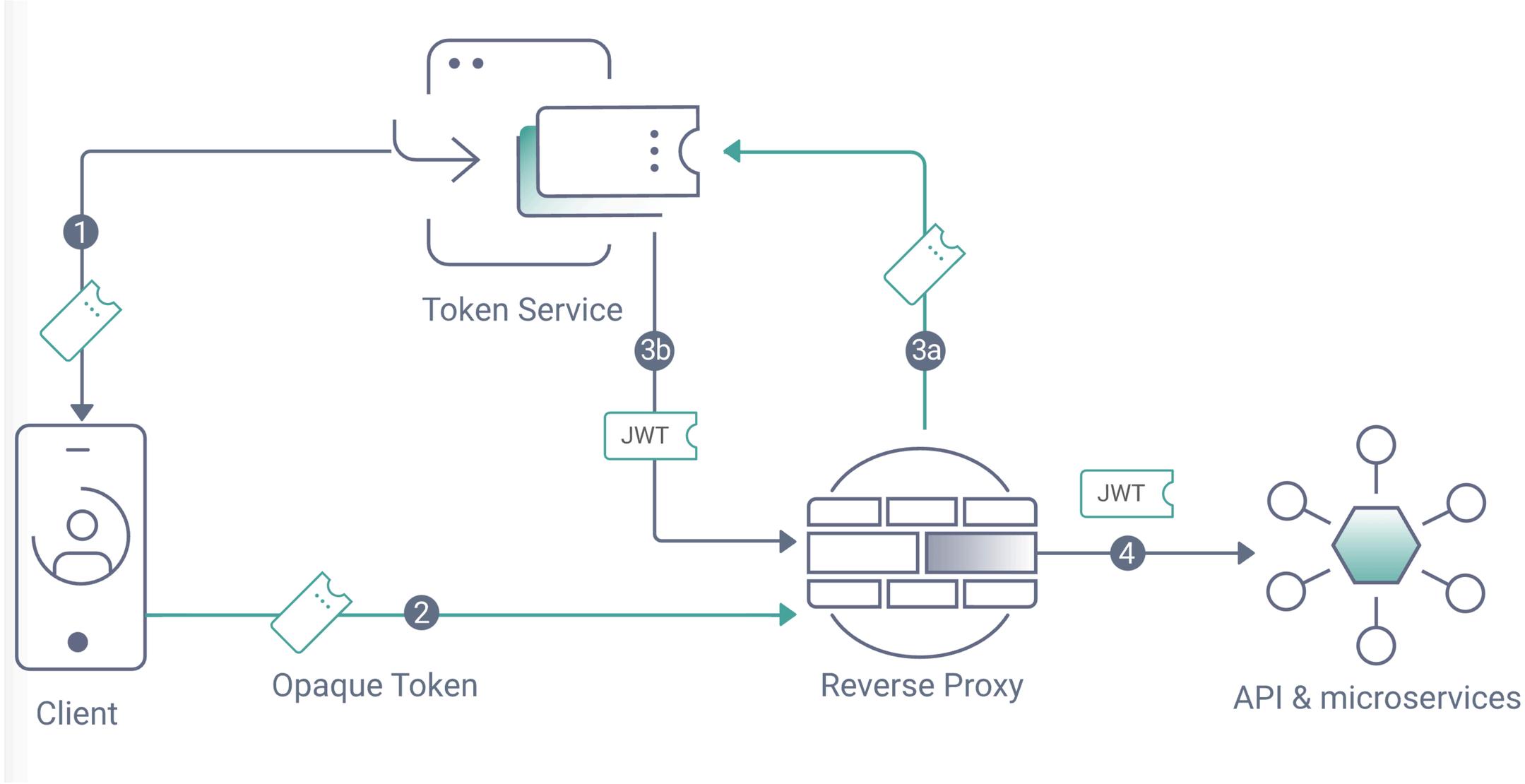

```
getUser:function(t){return
o.default.get("https://www.brewdog.com/uk/rest/uk/V1/customers/"+t,{headers:{'Cache-
Control':'no-cache, no-store, must-revalidate',Pragma:'no-cache',Expires:0,Authorization:"bearer
y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})},

getUserWithUsername:function(t){return
o.default.get("https://www.brewdog.com/uk/rest/uk/V1/customers/search?searchCriteria[filterGro
ups][0][filters][0][field]=email&searchCriteria[filterGroups][0][filters][0][value]="+t+"&searchCriteria
[filterGroups][0][filters][0][conditionType]=equals",{headers:{'Cache-Control':'no-cache, no-store,
must-revalidate',Pragma:'no-cache',Expires:0,Authorization:"bearer
y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})},

setMyLocal:function(t,s,n){return
o.default.put("https://www.brewdog.com/uk/rest/uk/V1/customers/"+t.id,{customer:{id:t.id,group
_id:t.group_id,email:t.email,firstname:t.firstname,lastname:t.lastname,store_id:t.store_id,website_i
d:t.website_id,custom_attributes:[{attribute_code:'my_local_id',value:s},{attribute_code:'my_local_
reset_date',value:n}]}},{headers:{Authorization:"bearer y99a5p6dhqspwr51h5z9r6h7t0zuaw5x"}})});
```

Your most sensitive endpoints are **authentication and password reset** endpoints.

# Need Application Authentication

- Service Mesh gives micro service to microservice communication policies
- Knows nothing about the API traffic itself.
- Need to design how we are going to pass authentication information and authorization information to downstream services.

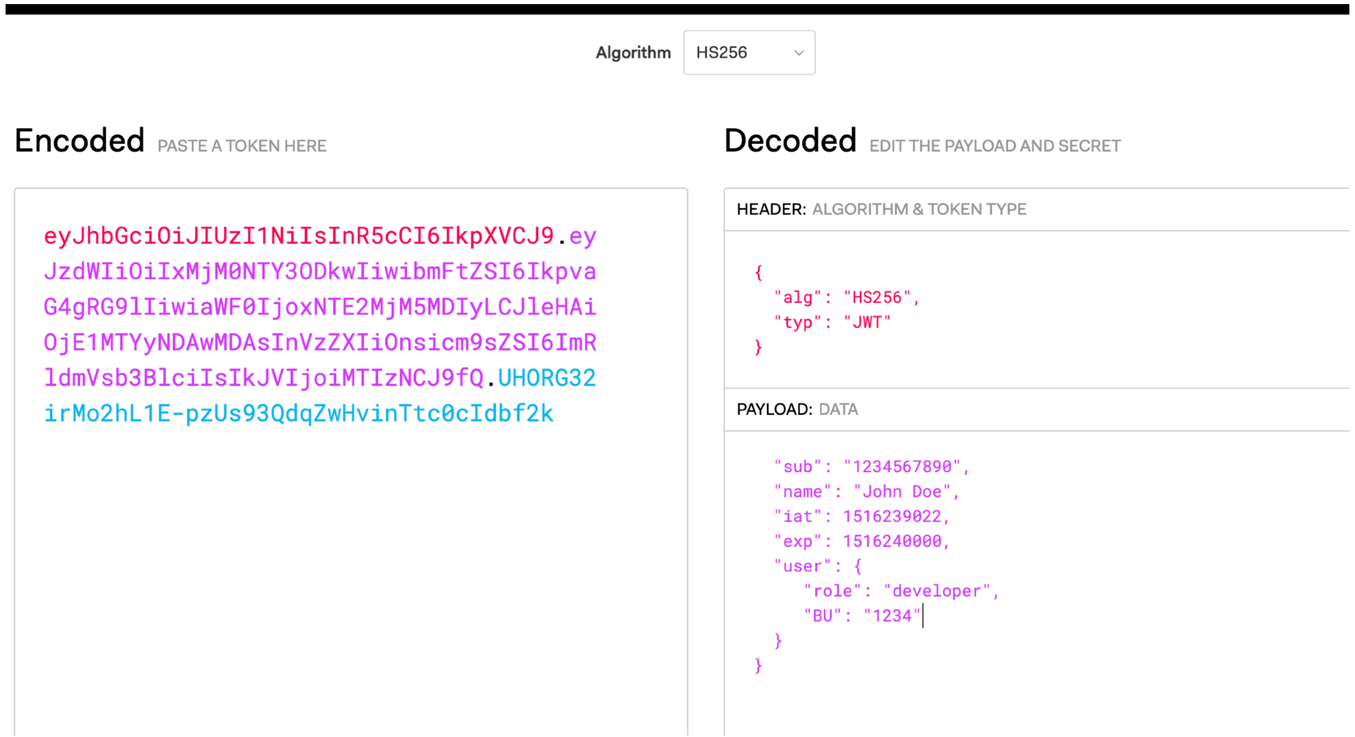


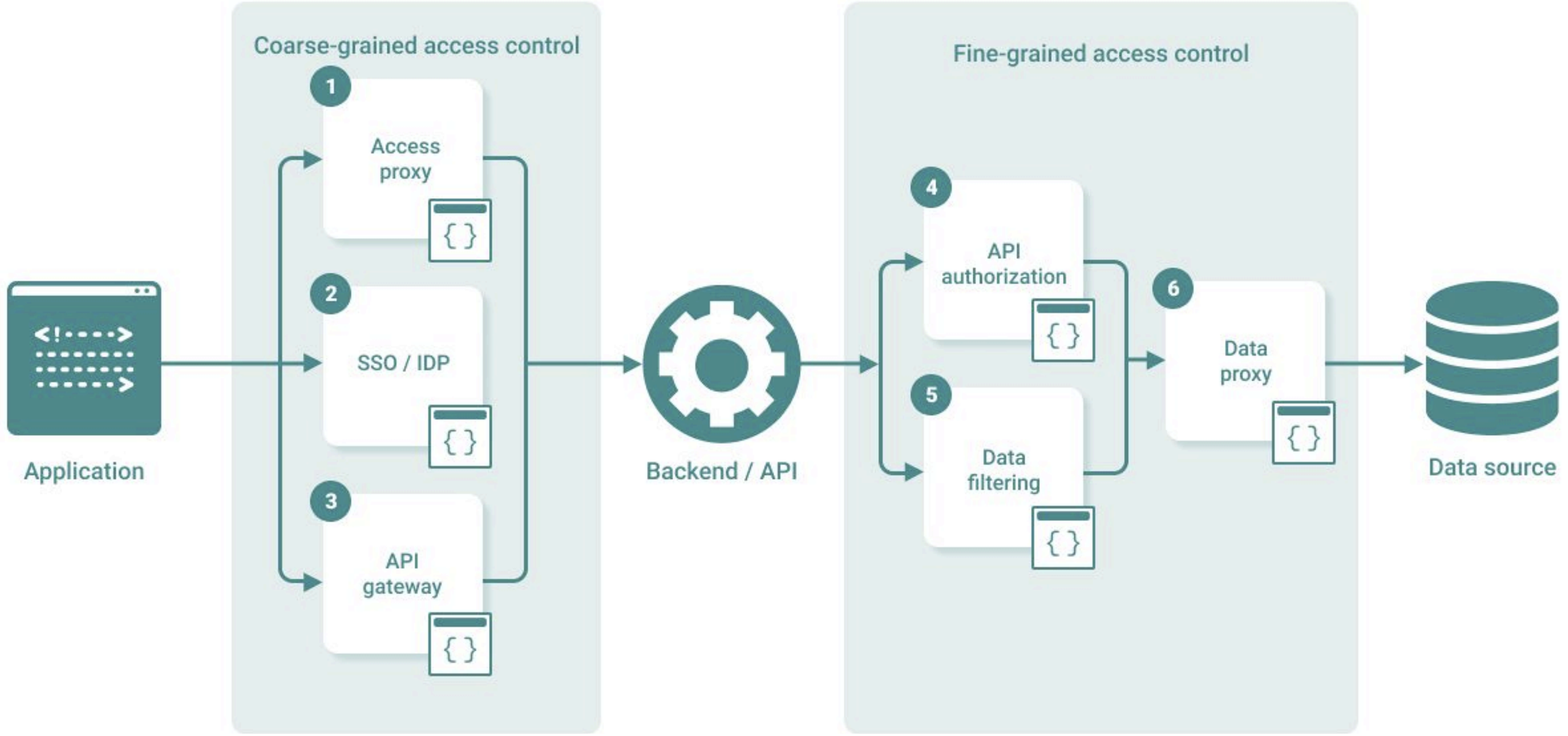



Phantom Token Pattern

# JWT Claims for Authorization

- Common pattern is to consume JWT claims to take authorization decisions
  - User role
  - Partner tier level
- **Before** anything else
  - Check the JWT contents against a JSON schema, both headers and claims
  - Token services/libraries will **most likely** test the obvious, but not custom data.
  - See **RFC 8725** for a complete check list



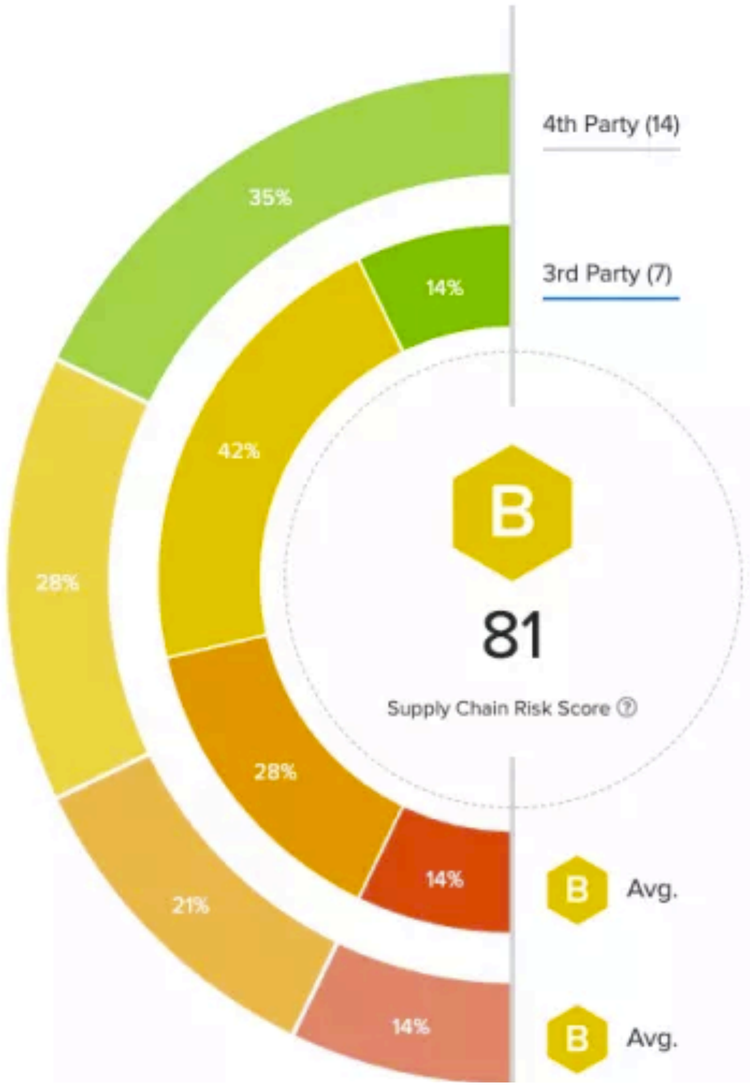



Authorization at every layer

In general, don't do this in the API/App code!


OVERWHELMED BY THE
AMOUNT OF MOVING PARTS ?


YOU NEED
PROACTIVE
SECURITY

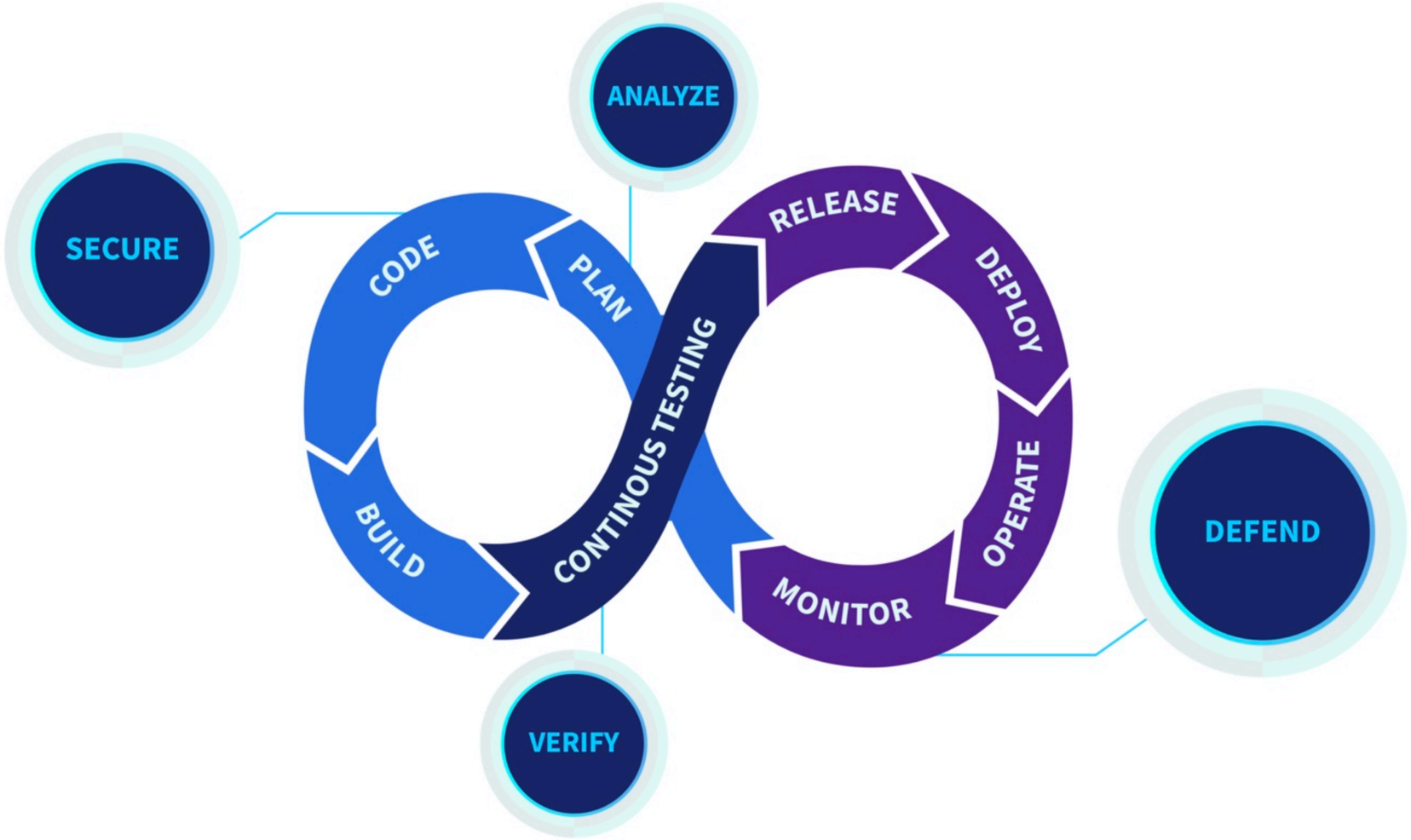



Columns Filters Search connections

8 Send Atlas Questionnaire Export Add to portfolio

| Company | Score | Party | Linkage Type |
|---|---|---|---|
| MyGRC | D 65 | 3rd | Web Crawl 43 days ago |
| EverLegal | C 79 | 3rd | Web Crawl 43 days ago |
| CloudHosting LLC | C 79 | 3rd | Web Crawl 43 days ago |
| PeopleOps Tools Inc | B 80 | 3rd | Web Crawl 43 days ago |
| OvoMarketing | B 85 | 3rd | Web Crawl 43 days ago |
| Acme Corporation | B 89 | 3rd | Web Crawl 43 days ago |
| Globex Corporation | A 90 | 3rd | Web Crawl 43 days ago |

Automated vigilantes!




Security embedded in API Lifecycle

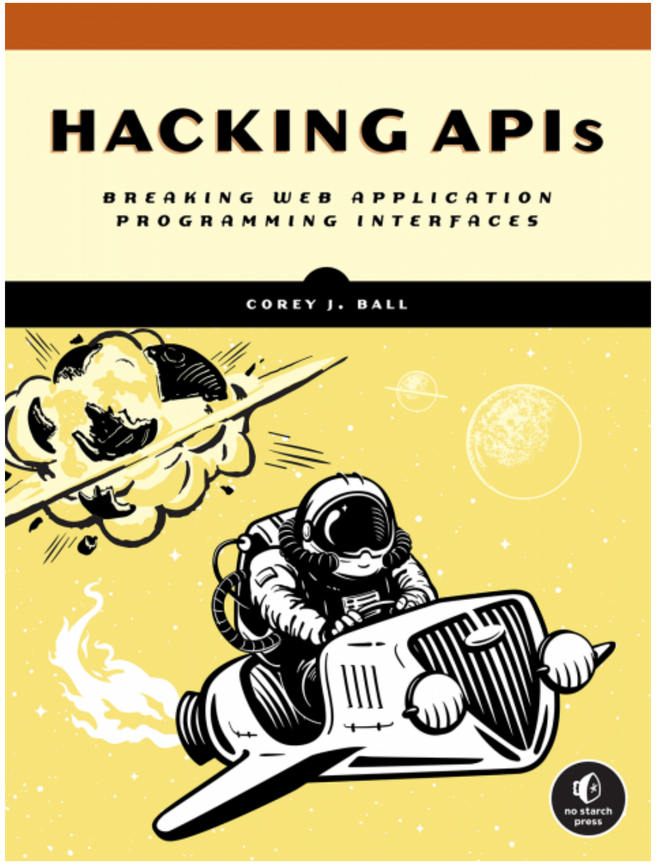

Code and Slides available at: https://github.com/isamauny/secappdev2023

# Learning more

## APISecurity.io

https://apisecurity.io/

## “Hacking APIs” - Corey Ball




https://nostarch.com/hacking-apis

## Learning Application Security


Buy the book