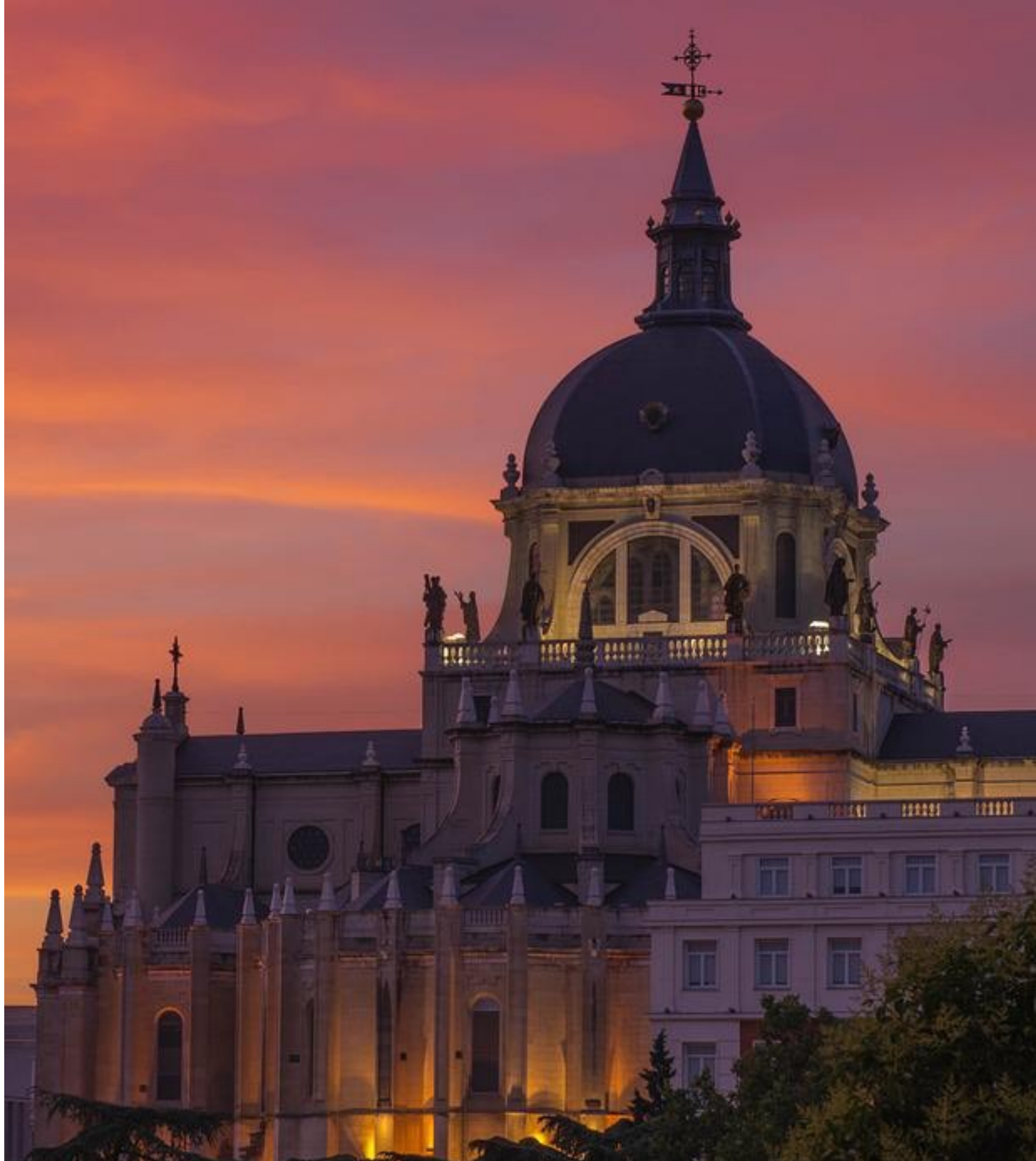
The (bright) future of API Security

Isabelle MAUNY - Field CTO - 42Crunch

Glad to be here!

- Field CTO / Founder of 42Crunch
- French National, living in Spain for past 20 years
- Most of career in the integration world, pioneering what would become API Management
- Fell quite recently into security.. but we will talk about that later.



APIs connect the world

Cloud

Healthcare

Banking

IoT

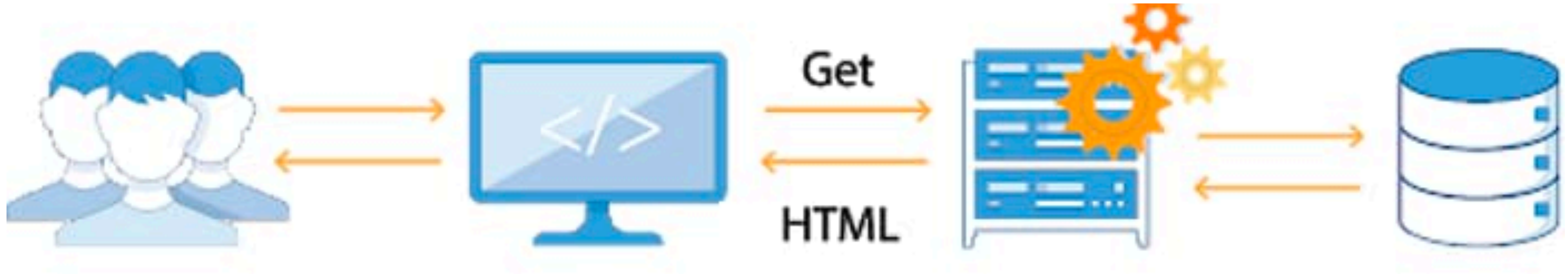
5G

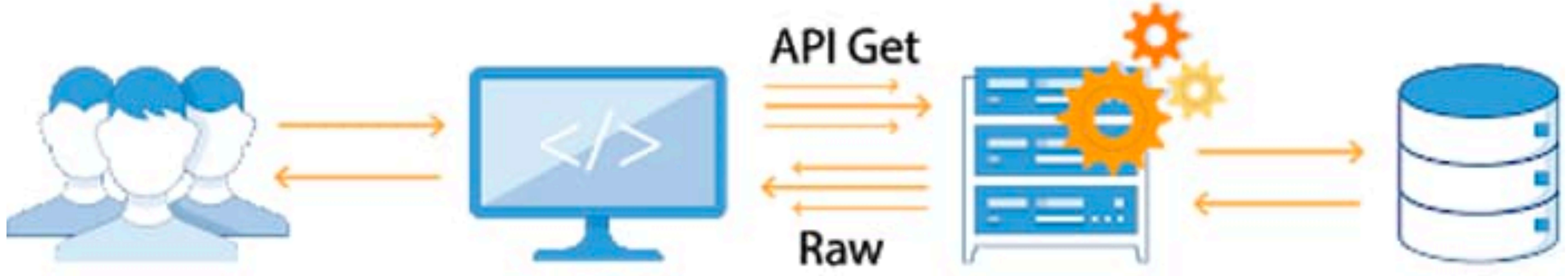
Data is the new gold!

APIs are a critical path to data

Equifax
Experian
Verizon
T-Mobile
Facebook
LinkedIN
Parler

WHY ARE APIS SUCH A PROBLEM?





Evolution of web architectures

We lost the server-side controller layer

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

Security Architecture has to evolve from protecting this…

…to protecting this!

***"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*

Development plays hard to catch…

APPLICATION
DEVELOPMENT

APPLICATION
SECURITY

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

Security is still an **afterthought**!

No news there.

Application Security is **hard**!

For everyone.

Too much to master ?

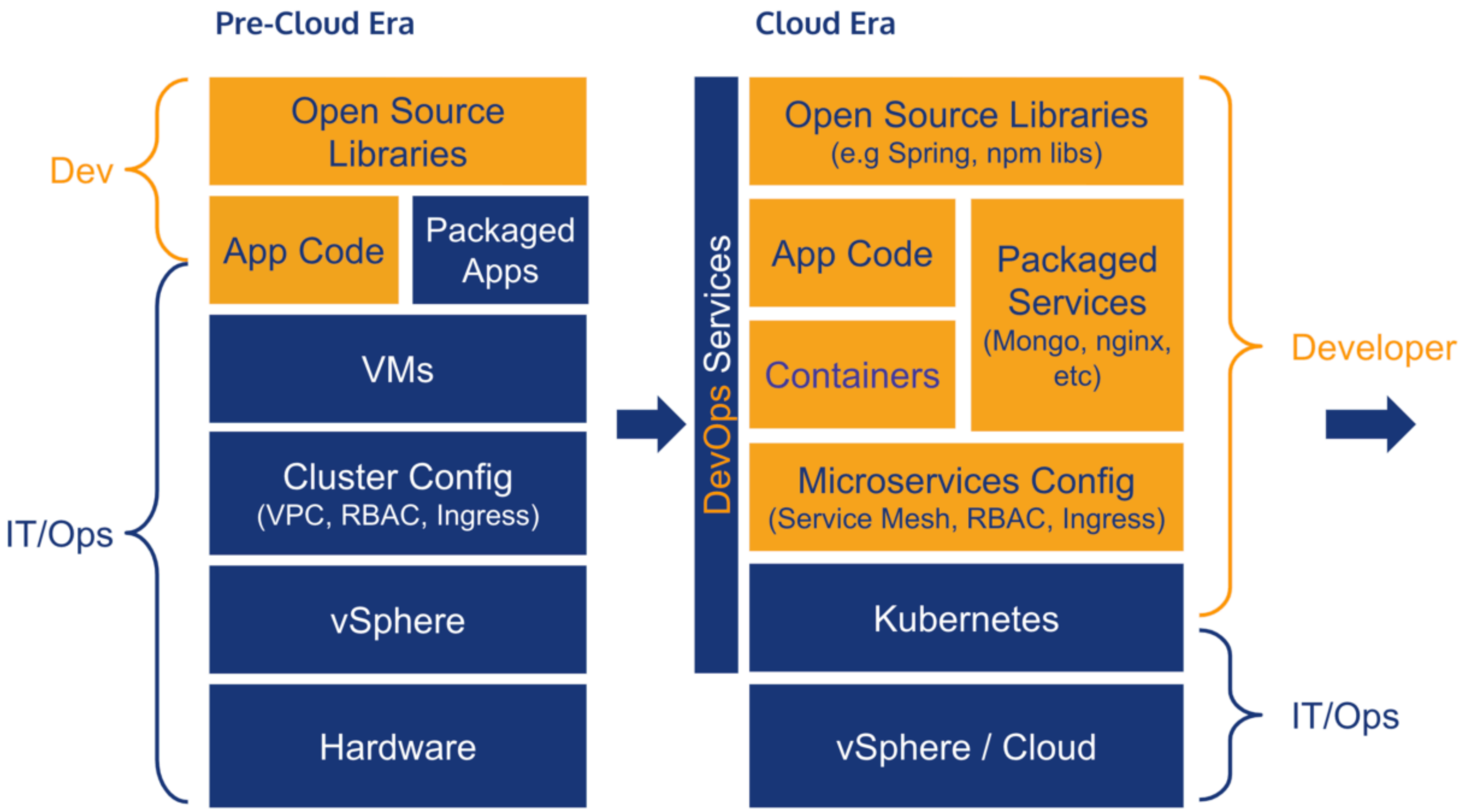
I am learning every day!

Thou shalt…

- Current security processes/tools create a lot of work for developers
  - False positives
  - Delayed builds (hours)
- Imposing security rules that impact productivity only results in friction, “malicious obedience” and frustration

Design Flaws

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


Pre-Cloud Era
Cloud Era
Dev
IT/Ops
Open Source Libraries
App Code
Packaged Apps
VMs
Cluster Config (VPC, RBAC, Ingress)
vSphere
Hardware
DevOps Services
Open Source Libraries (e.g Spring, npm libs)
App Code
Containers
Packaged Services (Mongo, nginx, etc)
Microservices Config (Service Mesh, RBAC, Ingress)
Kubernetes
vSphere / Cloud
Developer
IT/Ops

The AppSec stack

Increased role/responsibility of developers.

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

APIs have different vulnerabilities (REST)

- 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

CHEAT SHEET

OWASP

API Security Top 10

DOWNLOAD

🟦 Data Protection 🟧 Auth / Authorization 🟩 Governance/Operations

Parler (January 2021)

- Wild combination of issues!
- 70 TB of user’s data leaked
- Core Issues
  - Sequential IDs (IDOR/BOLA)
  - No Authentication
  - No Rate limiting
  - Leaked raw metadata about posts, including location
  - Deleted data was not deleted, just hidden in the UI

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

API1

API2

API3

API4

API5

API6

API7

API8

API9

API10

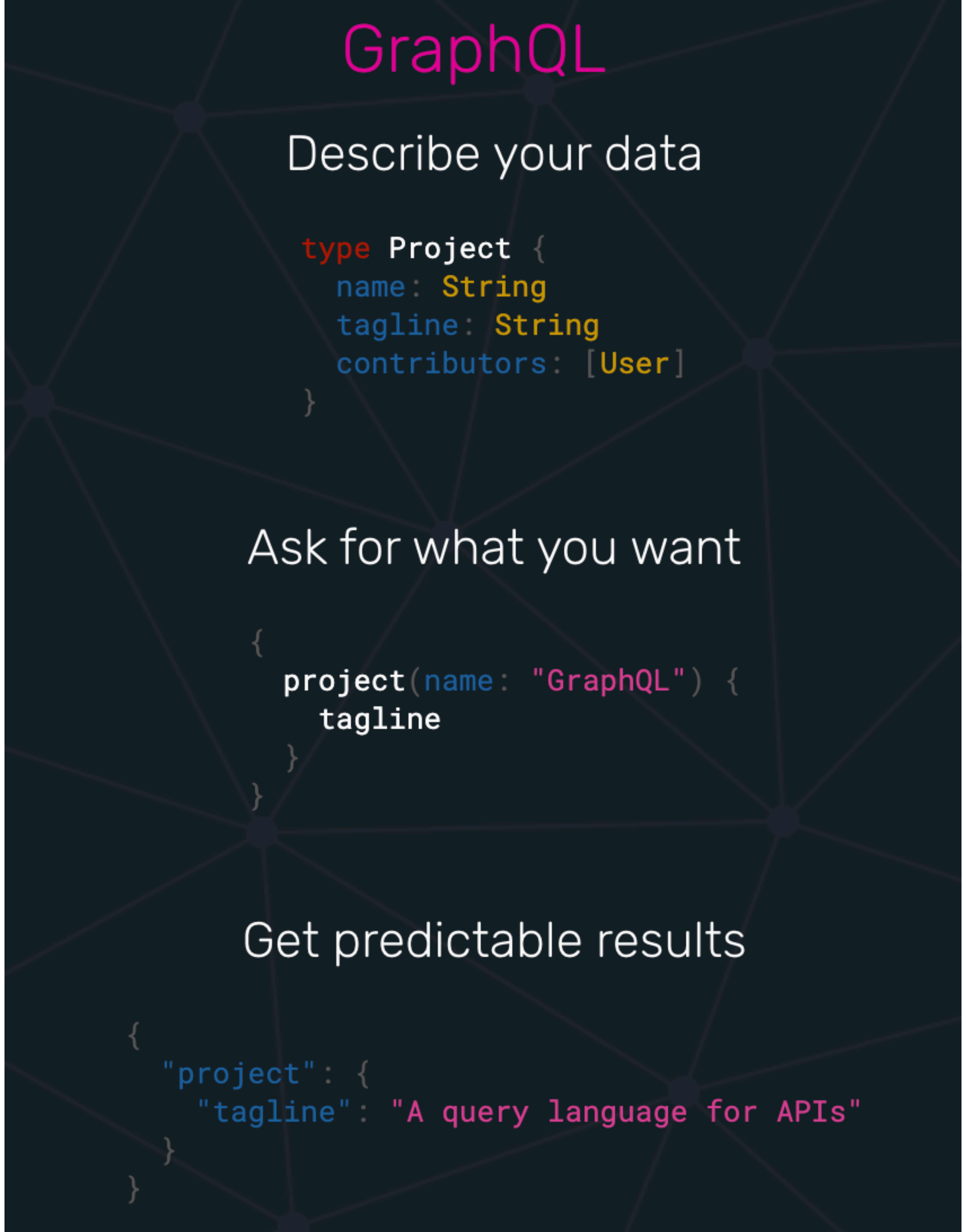
GET/accounts/id1/financial_info
GET/accounts/id2/financial_info

Zoom on BOLA

The #1 issue today

GraphQL

- Similar security issues as REST plus:
  - Queries complexity (DOS)
  - Queries recursivity (DOS)
  - Queries “suggestions”
- Authorization layer is complex as not covered by framework - Likely to led to more BOLA-style issues.



```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

How do we address this?

- Common language across Dev and AppSec
- Empower Developers
- Trust but Verify

- Cover security basics
- Restore Controller Layer
- Frameworks for core tasks

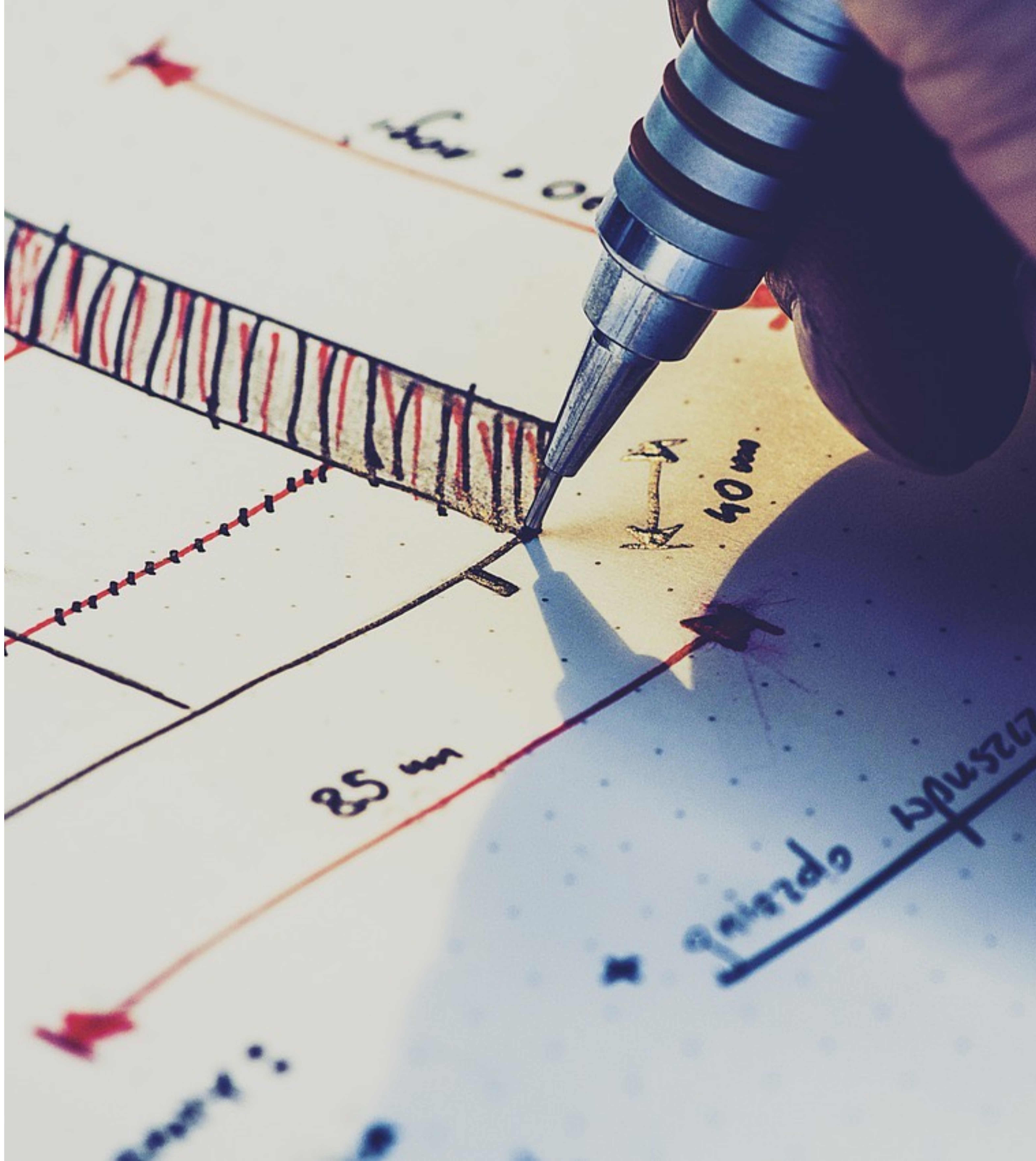
- Automation

Better Communication

- APIs are “popping up like mushrooms”
- AppSec teams usually have very limited knowledge/visibility about APIs development
- AppSec is shooting in the dark to find issues
- AppSec and Dev need a **common language** to describe those APIs

Common Language

- API blueprint is required
- Specifications like OpenAPI or AsyncAPI have a key role to play
- Why ?
    - Standard, Extensible language widely used by both sec and dev tooling
    - Enables Security as Code approach
    - Enables static analysis
    - Enables dynamic testing
- Enables **positive** security model



Access Allowed
by default

Block access for
suspicious traffic

Threats centric

Negative Security Model
(Deny List)

Access Denied by default

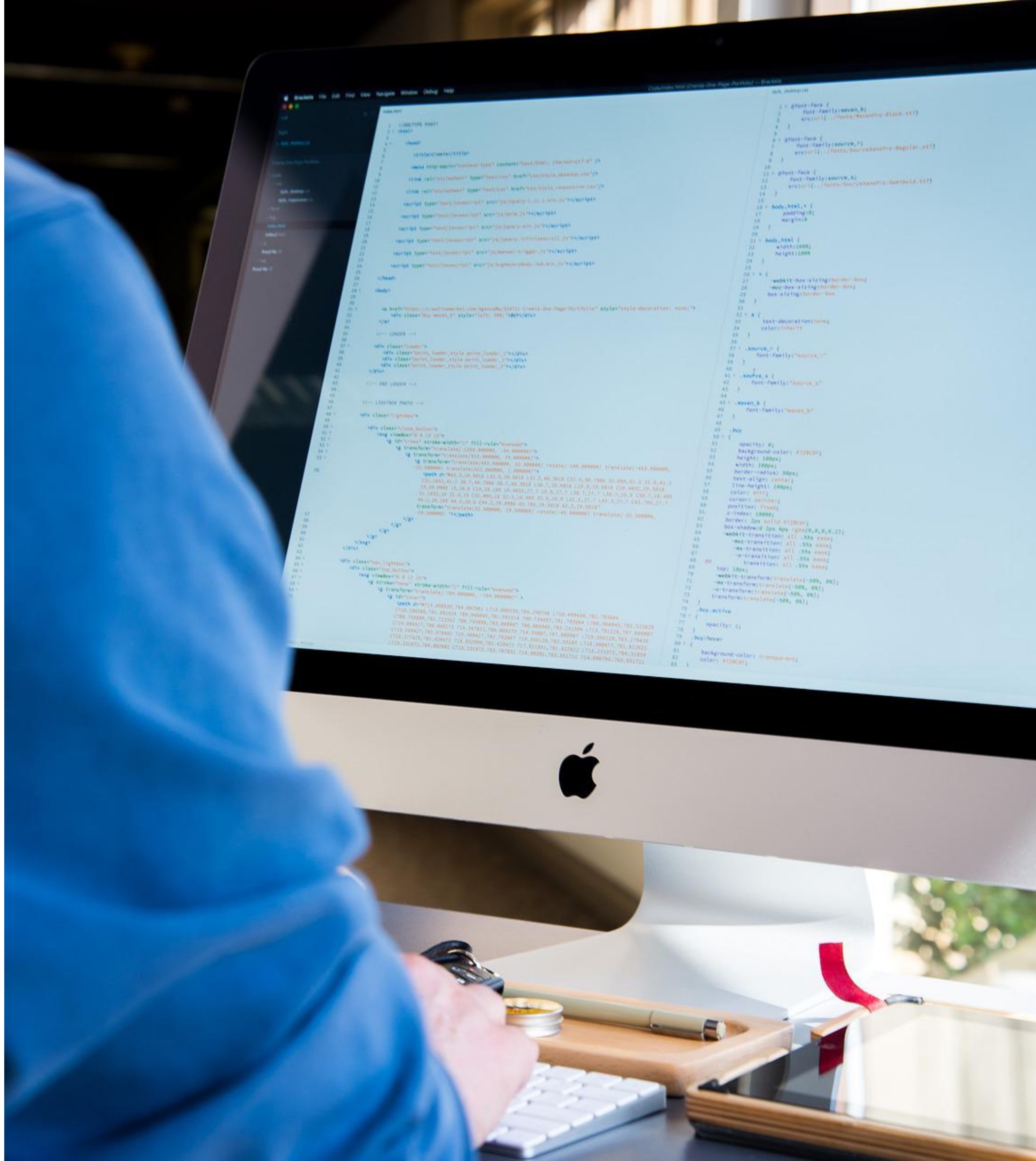
Allow Access only to approved traffic

Trust centric

Positive Security Model (Allow List)

Empower Developers

- “No shame, no blame”
- Tools which can be used from dev flow
    - Limited false positives
    - Easy to use from IDEs
    - Provide remediation guidelines
    - Interactive Security Testing



Controller Layer

- Control everything **server-side**
- Handles auth and authorization
  - Who has access to what, at operation **and** data level.
  - Who can talk to who ?
- Service Meshes/API Gateways play **part** of that role but we need more (especially for authorization / BOLA prevention)

Trust but Verify

- App Sec teams want to ensure corporate security standards are respected
- Allow them to express static rules of what is acceptable or not, for example:
    - OAuth with azn_code is mandatory
    - JWTs must be signed with PS256
    - All inbound parameters must be constrained by patterns and limits
- **Results visible to dev teams as early as possible.**

Cover the basics!

- Threat Modelling for APIs
- Input validation
  - Anything coming in: headers, body, query params, JWTs contents, etc.
- Output validation
  - Control the data: PII, Sensitive Data, tokens
- “Proper” rate limiting
  - By operation
  - Auth / Tokens endpoints
- Logging

What we see is working

- Educate developers
- Separate security controls so that development focuses on business logic
    - Authentication/Authorization
    - Input/Output Validation
- Provide corporate libraries for key functionality
    - Logging especially
- Prevent uncontrolled access to npm, DockerHub and similar.
- Create many "negative tests" (10X more than "200 OK" tests

Automate Security

Only solution with 1000’s of APIs to protect.

What we see is working

- Empower Dev teams with CI/CD templates they manage themselves
  - Particularly for large enterprises
- Automate “negative tests” for each release (even if it happens every day!)
- Automate “basic” pen-testing
- Protect the software supply chain by systematically validating libs and images
- Automate the injection of security policies

Future of API Security

- Dev and AppSec reconcile their conflicting goals through new processes and tools
- Developer are empowered to discover and fix security issues in their IDEs
- Security is expressed as code and automated