

CyberSecurityPanel

edycja #32

Już za moment
zaczynamy...



CyberSecurityPanel

edycja #32



CyberSecurityPanel

16 CZERWIEC 10:00

API są wszędzie. Jak wdrożyć kompleksową ochronę wszystkich Twoich interfejsów API?

Dyskusje poprowadzą: Artur Holeczek, Bartosz Chmielewski

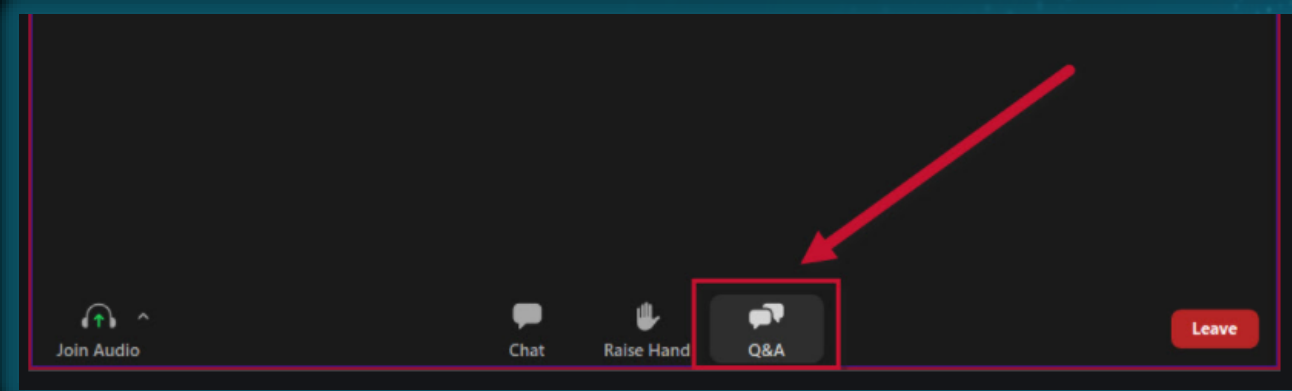
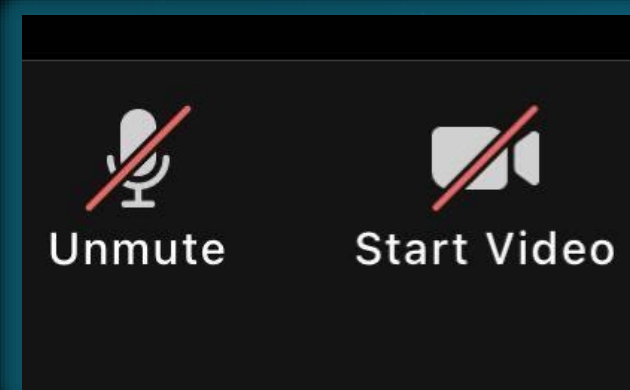
THALES
Building a future we can all trust

CLICO



CyberSecurityPanel

edycja #32



CyberSecurityPanel

problem



Nasze wyzwanie (czyli co i jak chcemy osiągnąć)

- Publikujemy obecnie dziesiątki tysięcy API. Nie mamy nad nimi takiej kontroli jakbyśmy chcieli.
- Interfejsy są publikowane przez różne zespoły i w różnych miejscach (on-prem, cloud).
- Używane są różne narzędzia, które nie są ze sobą połączone.
- Niektóre interfejsy API są krytyczne i bezpośrednio podłączone do krytycznych baz danych.
- Nie mamy ochrony przed atakami typu business-logic jak BOLA.
- Jak zbudować pełną widoczność?
- Jak zabezpieczyć się przed nadużyciami?
- Jak zarządzać ryzykiem w kontekście API?
- Czy WAF mi wystarczy?
- Jak odciąć nieautoryzowanych klientów?

CyberSecurityPanel

rozwiązanie



Oto obecny obraz sytuacji

- API są **silnie rozproszone** w chmurze, K8s, serverless i systemach legacy
- Własność jest **zdecentralizowana** — każdy zespół sam tworzy i udostępnia API
- Wiele punktów wejścia: API Gateways, service mesh i wywołania bezpośrednie
- **Rozrost API** jest powszechny: duplikaty, zapomniane i ukryte API
- **Brak jednego źródła prawdy** dla wszystkich API i ich użycia
- Nadzór jest niespójny: auth, limity i wersjonowanie zależą od zespołu
- Widoczność bezpieczeństwa jest rozproszona między narzędziami i środowiskami

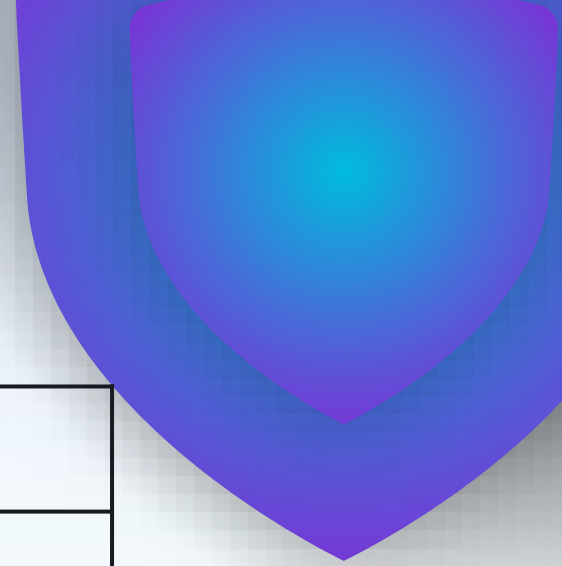


Obecnie mam API Gateways oraz API Management Czy to wystarczy?

- API Gateways and API Management only control known and registered APIs
- They enforce rules, policies, and traffic controls, not real-world usage behavior
- They have limited visibility into shadow, unmanaged, or forgotten APIs
- Security is based on declared configuration, not actual runtime reality
- Result: blind spots remain in modern distributed API environments



API Gateways vs API Management vs API Security



Feature	API Gateway	API Management	Imperva API Security
Request routing	✓ Core	⚠ indirect	✗
Authentication (JWT/OAuth validation)	✓	⚠ policy setup	⚠ detects abuse/anomalies
Rate limiting / throttling	✓	⚠ configuration	⚠ detects/stops abuse
API lifecycle (design, versioning, deprecation)	✗	✓ Core	✗
Developer portal & onboarding	✗	✓ Core	✗
API discovery (incl. shadow APIs)	⚠ limited	⚠ partial	✓ Strong
Business logic attack detection (e.g., BOLA, Excessive Data Exposure, etc.)	✗	✗	✓ Core
Bot & automated abuse detection	✗	✗	✓ Core
Data exfiltration / sensitive data detection	✗	✗	✓ Core
Threat Intelligence and Technical Attack Prevention	✗	✗	✓ Core
Observability & analytics	⚠ basic logs	✓ strong dashboards	✓ security-focused analytics

Najistotniejsze zagrożenia...



Business Logic

1. Broken Object Level Authorization
3. Broken Object Property Level Auth
5. Broken Functional Level Auth

API Specific

2. Broken Authentication
8. Security Misconfiguration
9. Improper Inventory Management

Bot/Automated Attacks

4. Unrestricted Resource Consumption
6. Unrestricted Access to Sensitive Business Flows

Web Request/Response Related

7. Server-side Request Forgery
10. Unsafe Consumption of APIs

OWASP API Security Top 10

Unlike OWASP Web Security Top 10, most of the API Security Top 10 threats are caused by attacks targeting specific APIs exposing victim application's business logic vulnerabilities.

Bad actors use bots to scan and identify vulnerable APIs.

Successful attacks are automated and persistent, with well formatted API calls that evades detections of conventional security monitors.

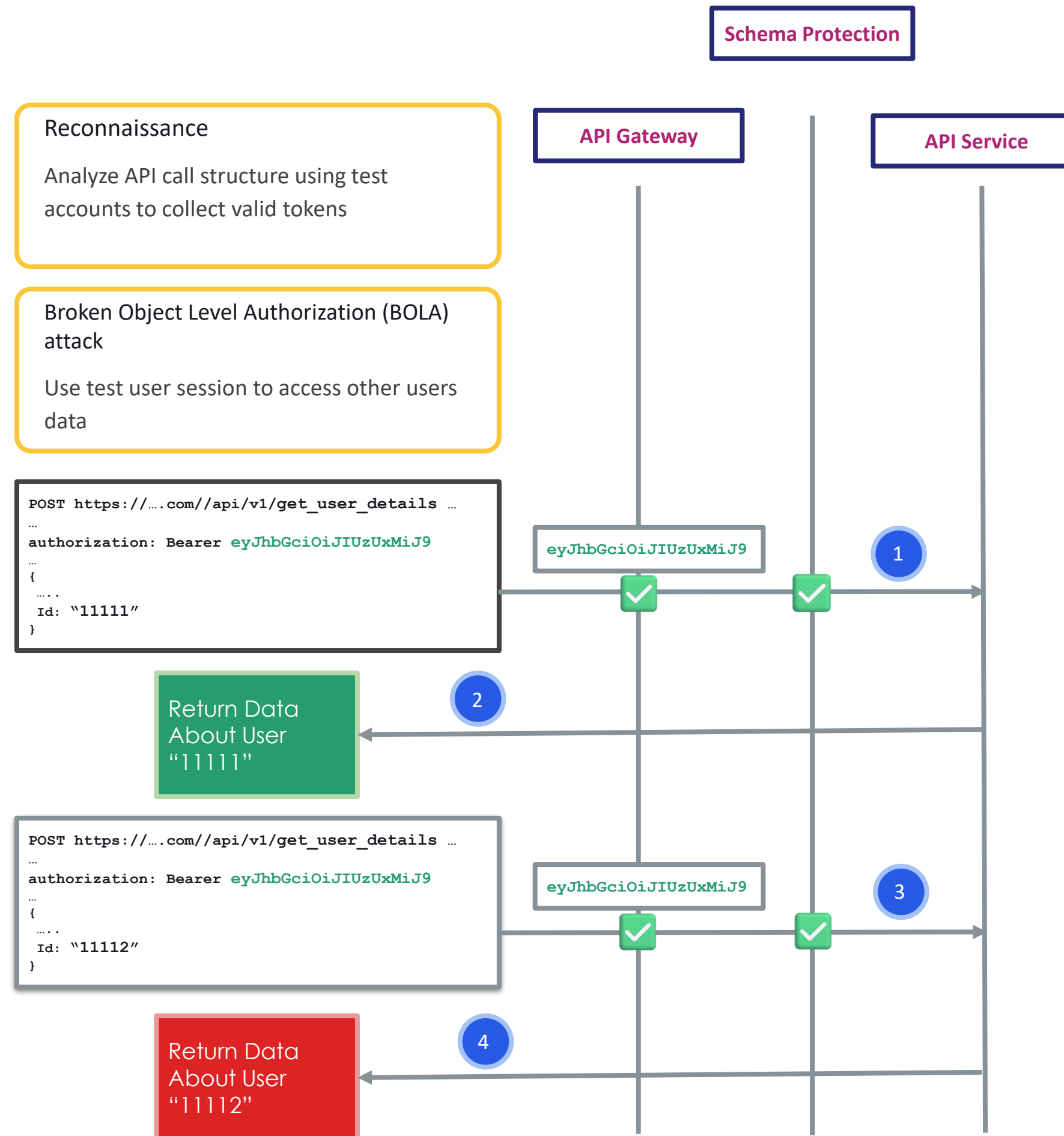
THALES

CYBERSECURITY

Building a future we can all trust



Jak wygląda atak BOLA?



Kiedy Schema Protection POWINNA być używana

WHERE SCHEMA PROTECTION FITS



Ideal for

- 1 Stable production APIs
- 2 Admin / privileged APIs
- 3 High assurance enforcement use cases
- 4 Legacy APIs with predictable behavior



Not ideal for:

- 1 Rapidly changing APIs
- 2 Large microservice environments
- 3 Shadow / unknown APIs
- 4 Business Logic Abuse

THE RIGHT APPROACH: REAL-TIME API SECURITY



- 1 **Real-Time Discovery**
Continuously identify new, changed and shadow APIs
- 2 **Behavioral Protection**
Detect BOLA, auth abuse, token misuse, data scraping, anomalies & more
- 3 **Risk-Based Prioritization**
Focus on exploitable & business critical risks first
- 4 **Selective Schema Enforcement**
Apply schema controls only where they create high value

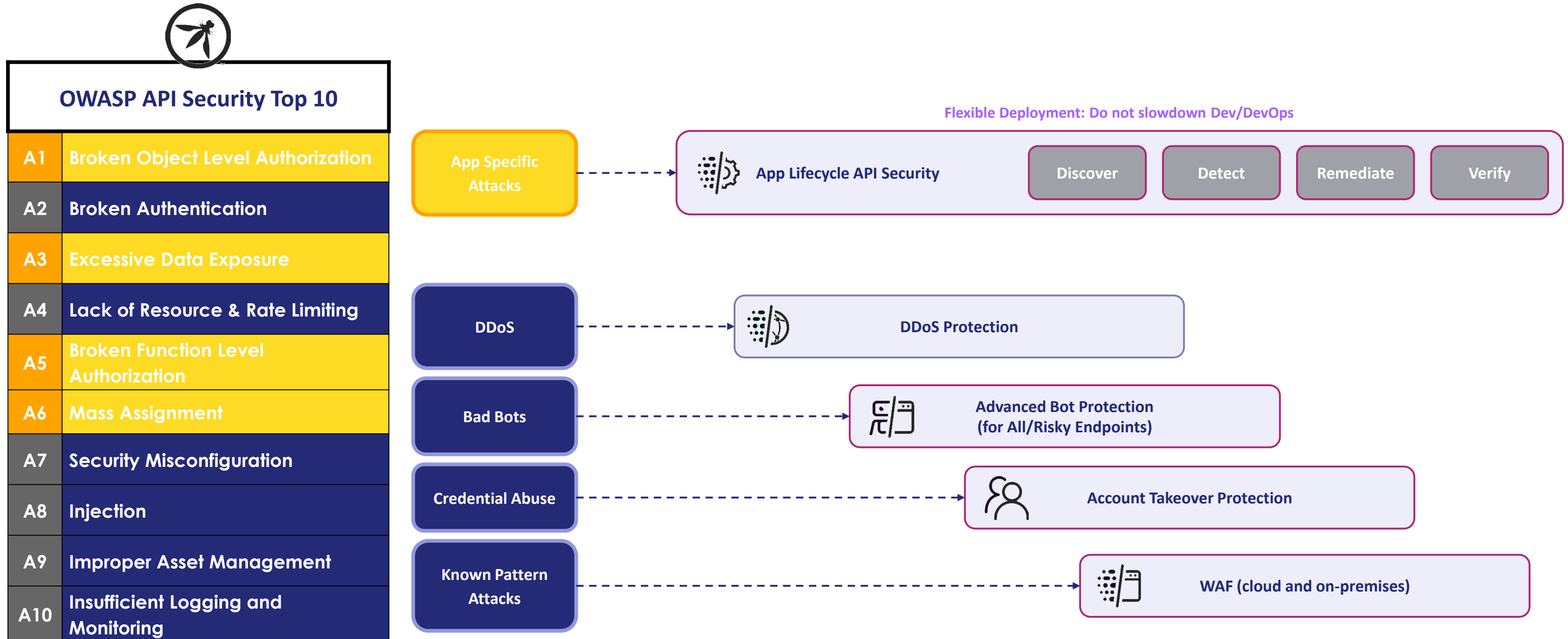
Co sprawia, że problem zabezpieczania API jest jeszcze większy?

- Sztuczna inteligencja w niepowołanych rękach (boty, wyrafinowane schematy ataków)
- Środowisko rozproszone/hybrydowe (=złożoność)
- Różne zespoły i zestawy narzędzi wykorzystywane do tworzenia i publikowania interfejsów API
- Dynamika

Spróbuj zapytać samego siebie...

- Czy wiem, ile interfejsów API mam obecnie?
- Które z nich są najważniejsze?
- Czy zastosowałem wobec nich wszystkie niezbędne środki bezpieczeństwa?
- Czy obowiązuje w odniesieniu do nich proces zarządzania bezpieczeństwem?

Jak zatem skutecznie chronić moje interfejsy API przed zagrożeniami?



Podstawowe funkcje systemu API Security

- Realtime API Discovery
- Data Classification
- API Risks Discovery
- API Risk prevention*
- Schema enforcement*

* Depends on deployment type



Model operacyjny API Security



Discover & Classify

Discover and classify APIs by criticality, ownership, sensitivity, and business function.



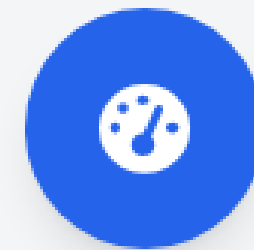
Identify Vulnerabilities

Understand Risk Exposure. Find and assess API vulnerabilities, misconfigurations, and abuse risks.



Prioritize Risks

Prioritize risks based on impact and likelihood and define targeted mitigation actions.



Risk Mitigation

Implement mitigations, track KPIs, and scale what works.



Optimize

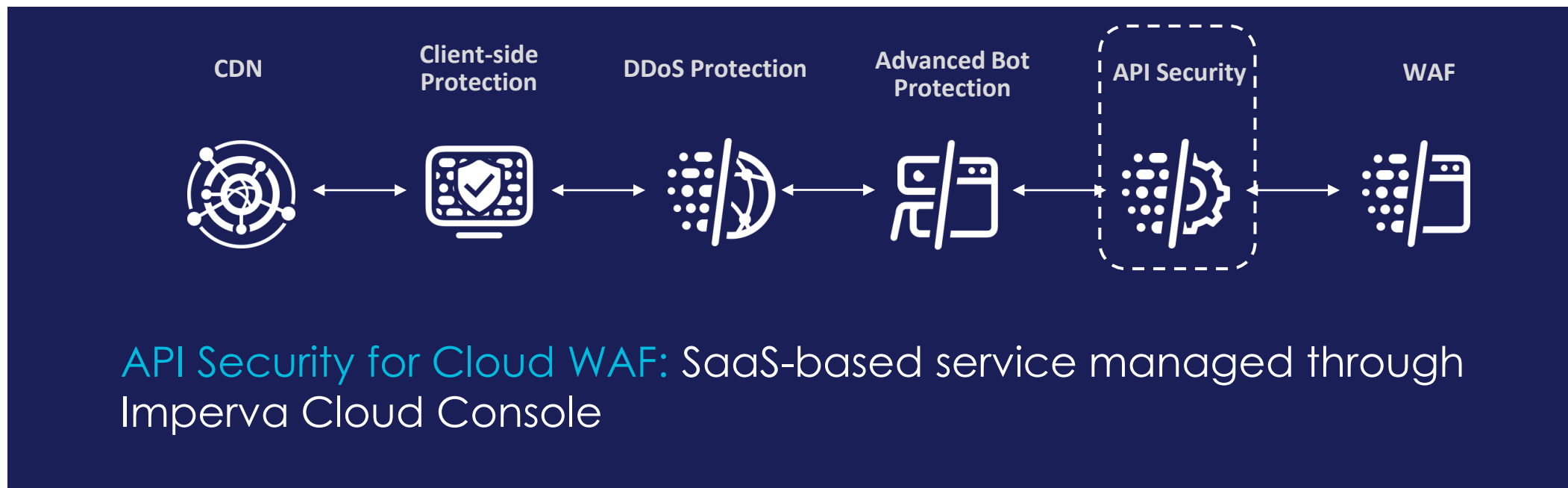
Automate workflows and policy enforcement to continuously improve and scale security.



SECURE APIS. REDUCE RISK. DRIVE BUSINESS FORWARD.

Deployment options

Cloud WAF Integrated



Supported Integrations

- WAF Gateway
- Kong
- MuleSoft
- Apigee
- NGINX
- F5 (HSL)

Anywhere



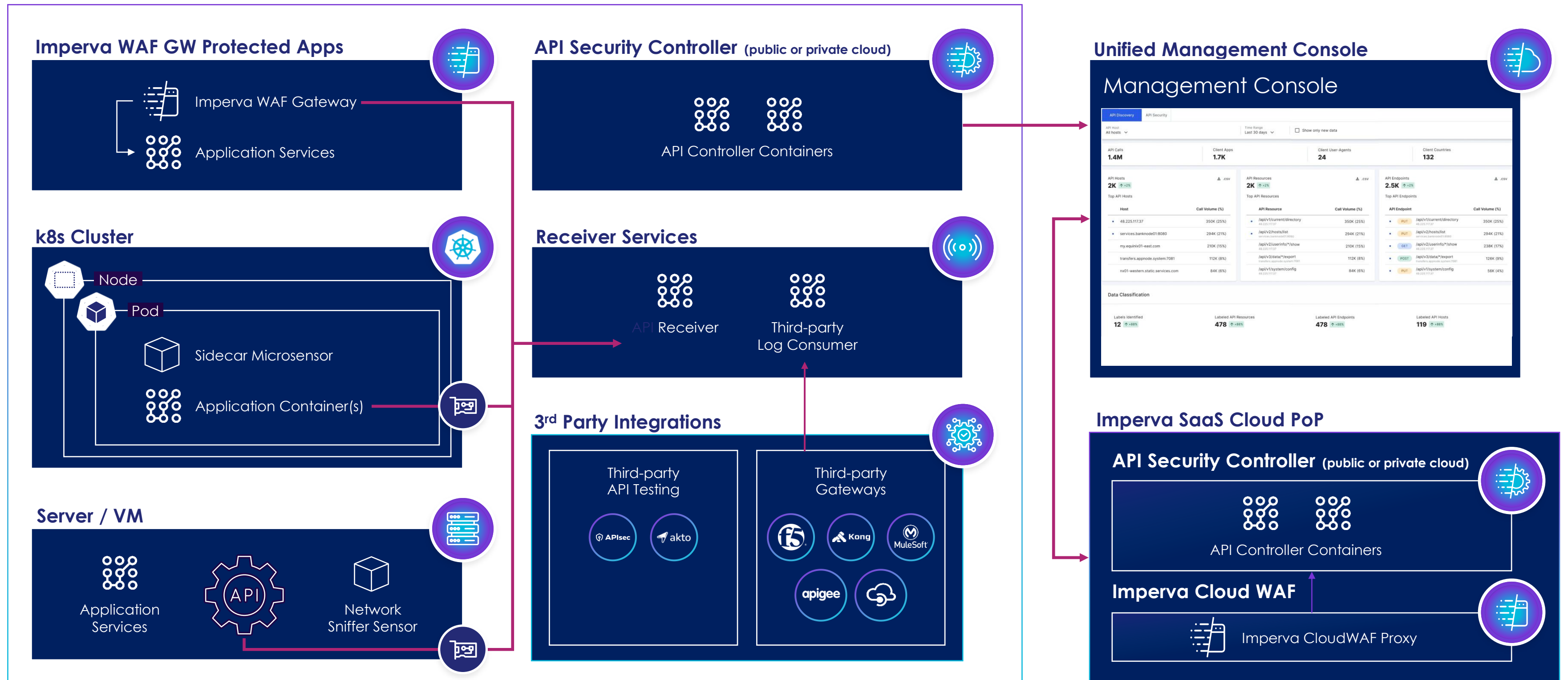
Sensor Options

- K8s sidecar sensor
- Runtime sensor
- OS Sensor
- SPAN Sensor

Console Options

- SaaS Console
- Standalone Console

API Security Anywhere Architecture.



CyberSecurityPanel

demo



API Discovery

POST /api/GetSum

Baselined | ID 2250143961 | Host lab.spm.pl

Tags

API Specifications | Vulnerabilities and Risks

Request | REST

Search

Type: All selected × | Required/Optional: Show all ▾ | Data labels: Show all ▾ | Label Progress: Done ×

Authentication Location: http-req-header-auth

^ POST Request details application/json

```
^ Header
  Content-Type: application/json
^ Cookies
  incap_ses_* String Optional
  nlbi_* String Optional
  visid_incap_* String Optional
^ body
  Object Required
  arg2 String Required
  arg1 String Required
```

Copy as JSON

Response

Search

Type: All selected × | Required/Optional: Show all ▾ | Data labels: Show all ▾ | Label Progress: Done ×

^ 200 HTTP Response Code application/json

```
^ Header
  Content-Type: application/json
^ body
  Object Required
  data Number Required
  status_message String Required
  status Number Required
```

Copy as JSON

API Data Classification

Edit data label

API Risks Identified

Labels for the Parameter
14 Total | **9** Sensitive | **5** Non-Sen

Filters

API Endpoint

- POST /api/GetUserData-... lab.spm.pl
- POST /api/GetProduct lab.spm.pl
- POST /api/ReflectInput lab.spm.pl
- POST /api/GetUserData lab.spm.pl
- POST /api/GetUserData-... lab.spm.pl

1 - 5 of 5 Rows per page 10

Validate

200 HTTP Response Code

application/json

```
^ Header
  Content-Type: application/json

^ body
  Object Required
  ^ zipcode
    Number Required
    String Required
    city String Required Done address:city-usa
    streetaddr String Required Done address:streetaddr
    email String Required
    dob Number Required
    gender String Required
    ssn String Required Done govtid:ssn-usa
    state String Required Done address:state-usa
    cc_number Number Required
    phonenum String Required
    first_name String Required Done name:firstorlastname
    status Number Required
    last_name String Required Done name:firstorlastname
    status_message String Required
    country String Required Done address:country
```

Copy as JSON

Risky API Endpoints

2 Total | **2** OWASP | **0** Others

Search .csv

Call Volume (%)

2K	0%
2K	<1%
2K	0%
2K	0%
2K	0%

1

API Threat Detection

API Risks Identified

Labels for the Parameter	Labeled API Endpoints	OWASP API Top 10 Risks	Risky API Endpoints
14 Total 9 Sensitive 5 Non-Sensitive	3 0%	1 0%	2 Total 2 OWASP 0 Others

System Defined Risks

Broken Object Level Authorization(BOLA) | OWASP API01 ⋮
Detected on Jan 9 2025, 09:01

Unique values per session across parameters

Parameter	Unique Values
id	6
category	10

Unique Parameter Values

■ Unique Values

i This endpoint is considered at risk of BOLA abuse because there is a high number of sessions, in which high number of API calls against the endpoint are showing a consistently low number of unique values for these parameters. This is an indicator that the parameters are likely corresponding to an object unique to the session. Any successful attempt for a different parameter value to obtain a different object can potentially be a BOLA abuse. ✕

API Threat Protection (new)

lab.spm.pl

address(4) govtid(1) + 1 more

Select Create policy

Detection type: BOLA | Traffic Source: CWF | Websites: 1 | Endpoints: 1

Settings | Endpoints

Endpoints

Filters

Search Use recommended

Define your own thresholds for susceptible BOLA parameters or use the recommended threshold. You can remove endpoints from the policy by selecting them and clicking "Remove endpoints". Any changes to the endpoints will only take effect after clicking "Create".

(-) <input type="checkbox"/>	API Endpoint	Website	Authentication Location	
^ <input type="checkbox"/>	POST /api/ReflectInput lab.spm.pl	lab.spm.pl	http-req-header-auth	

Susceptible Parameter	Type	Threshold ⓘ	Recommendation ⓘ
input Path: http-req-body->input	body	5	In progress

1 - 1 of 1 Rows per page 10

Kluczowe wnioski

- Interfejsy API wymagają solidnej strategii bezpieczeństwa.
- Sama zapora WAF to za mało.
- Sztuczna inteligencja jest Twoim sprzymierzeńcem (i wrogiem).
- Ataki na logikę biznesową są najbardziej niebezpieczne.
- Thales pomoże Ci zabezpieczyć Twoje interfejsy API.

CyberSecurityPanel

pytania i odpowiedzi



CyberSecurityPanel

