CyberSecurityPanel

edycja #32

Już za moment zaczynamy...

CyberSecurityPanel

edycja #32

THALES
Building a future we can all trust

CLICO

CyberSecurityPanel

16 CZERWIEC 10:00

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

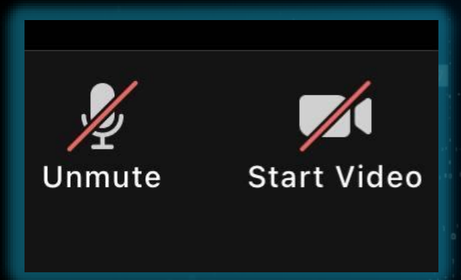
Dyskusje poprowadzą: Artur Holeczek, Bartosz Chmielewski

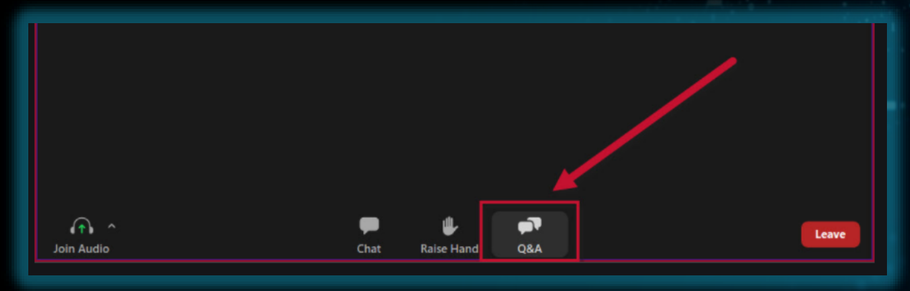
THALES
Building a future we can all trust

CLICO

CyberSecurityPanel

edycja #32





CyberSecurityPanel

problem

THALES
Building a future we can all trust

CLICO

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

THALES
Building a future we can all trust

CLICO

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

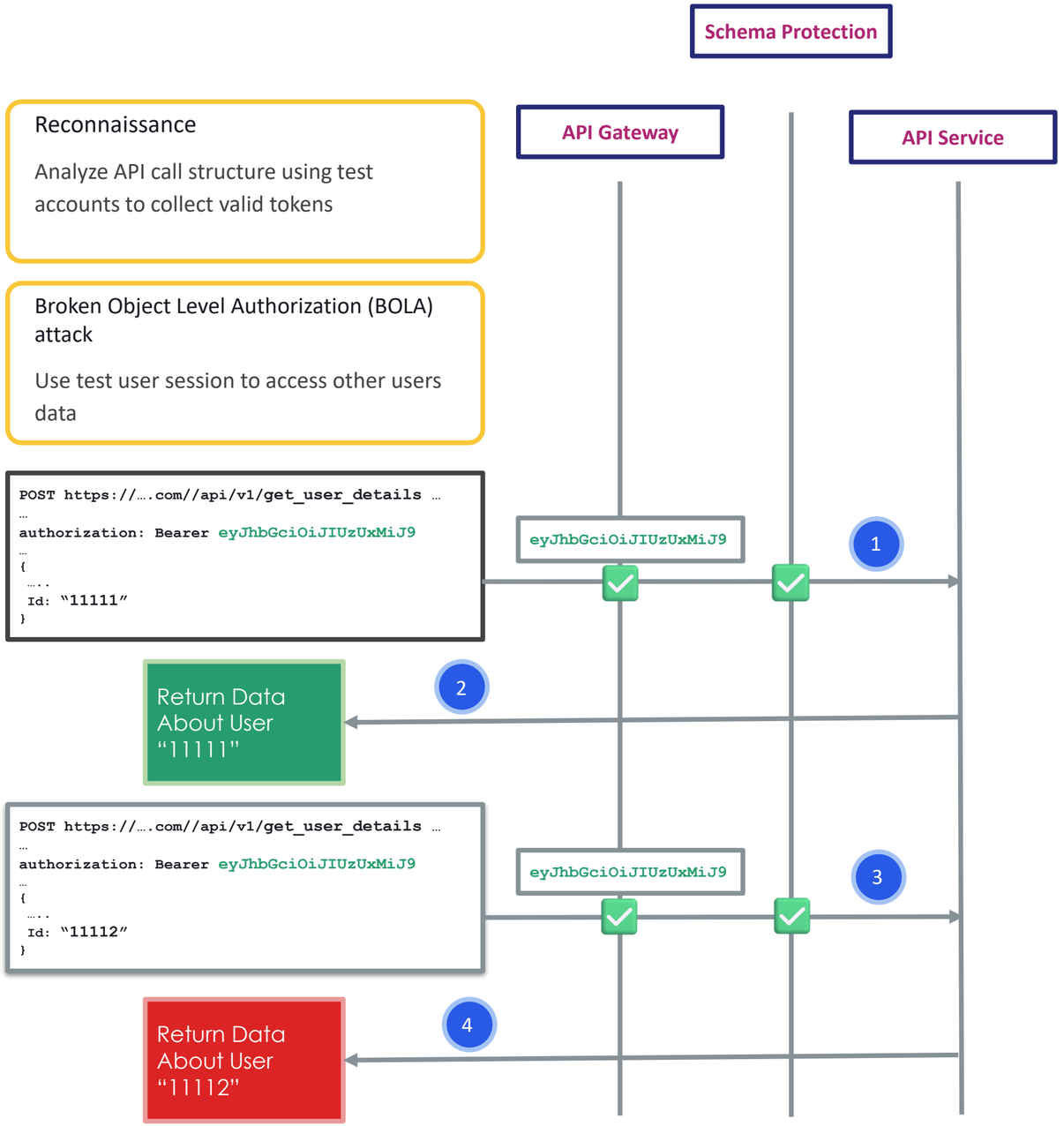
OWASP API Security Top 10

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.

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

THALES
CYBERSECURITY

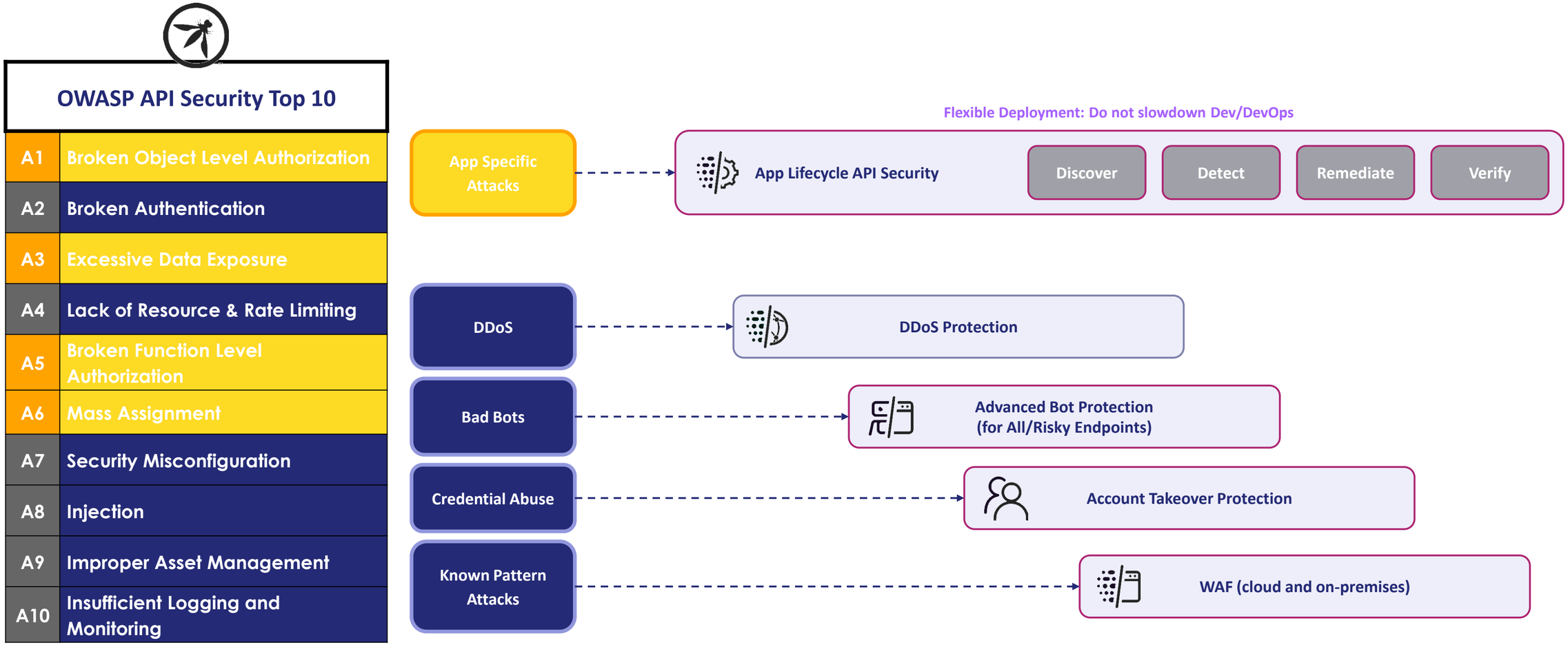
Co sprawia, że problem zabezpieczania API jest jeszcze większy?

- 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.

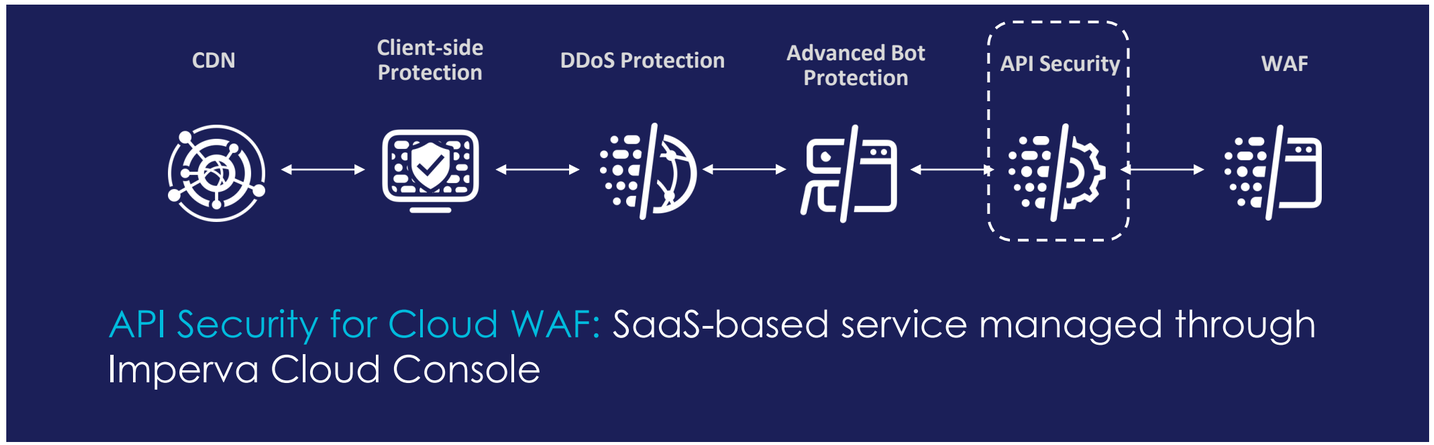
Optimize

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

SECURE APIS. REDUCE RISK. DRIVE BUSINESS FORWARD.

Deployment options

Cloud WAF Integrated



API Security for Cloud WAF: SaaS-based service managed through Imperva Cloud Console

Anywhere

API Security Anywhere: Customer-deployed and managed in the Imperva Cloud Console or Self-managed Local Console

Supported Integrations

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

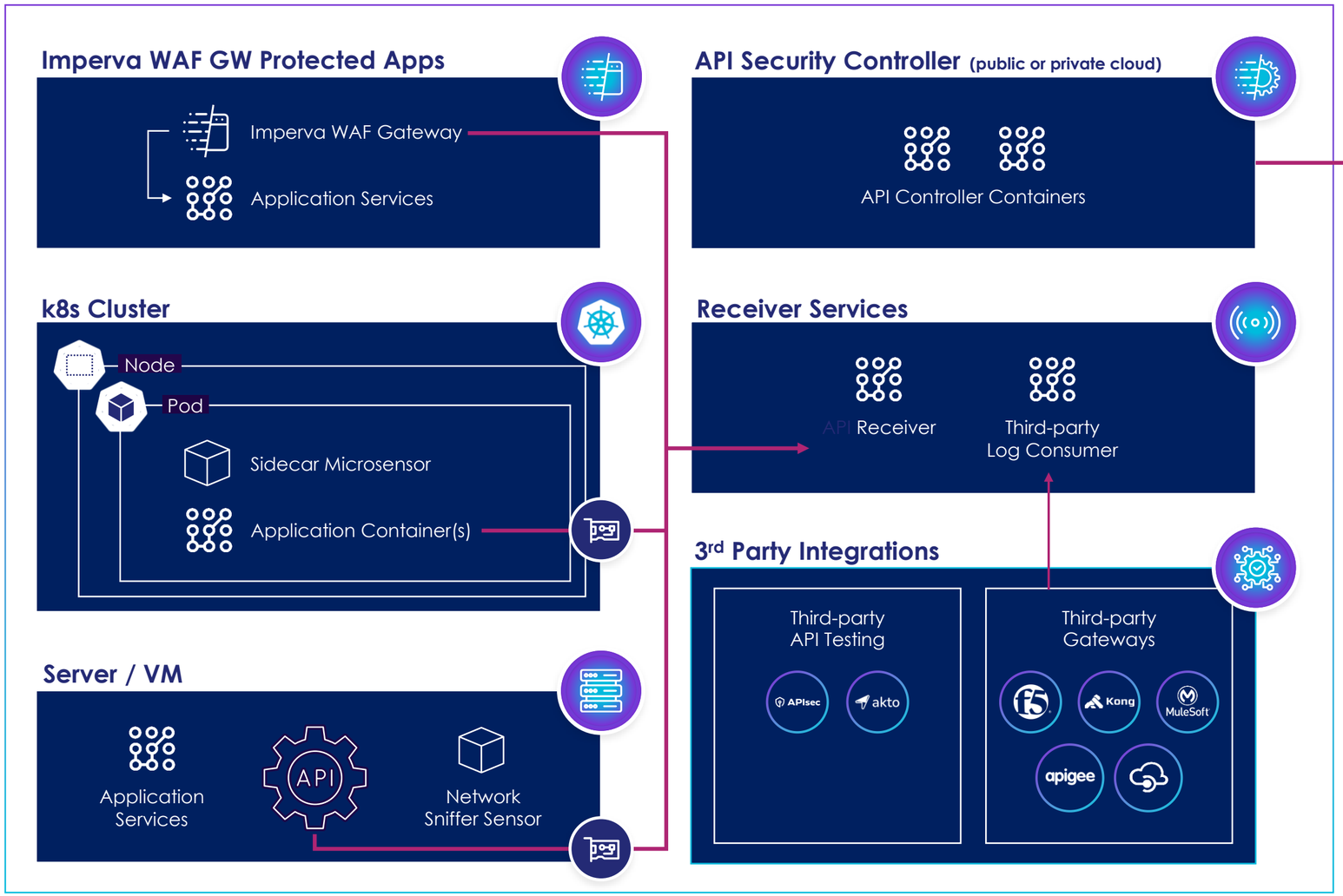
Sensor Options

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

Console Options

- SaaS Console
- Standalone Console

API Security Anywhere Architecture.



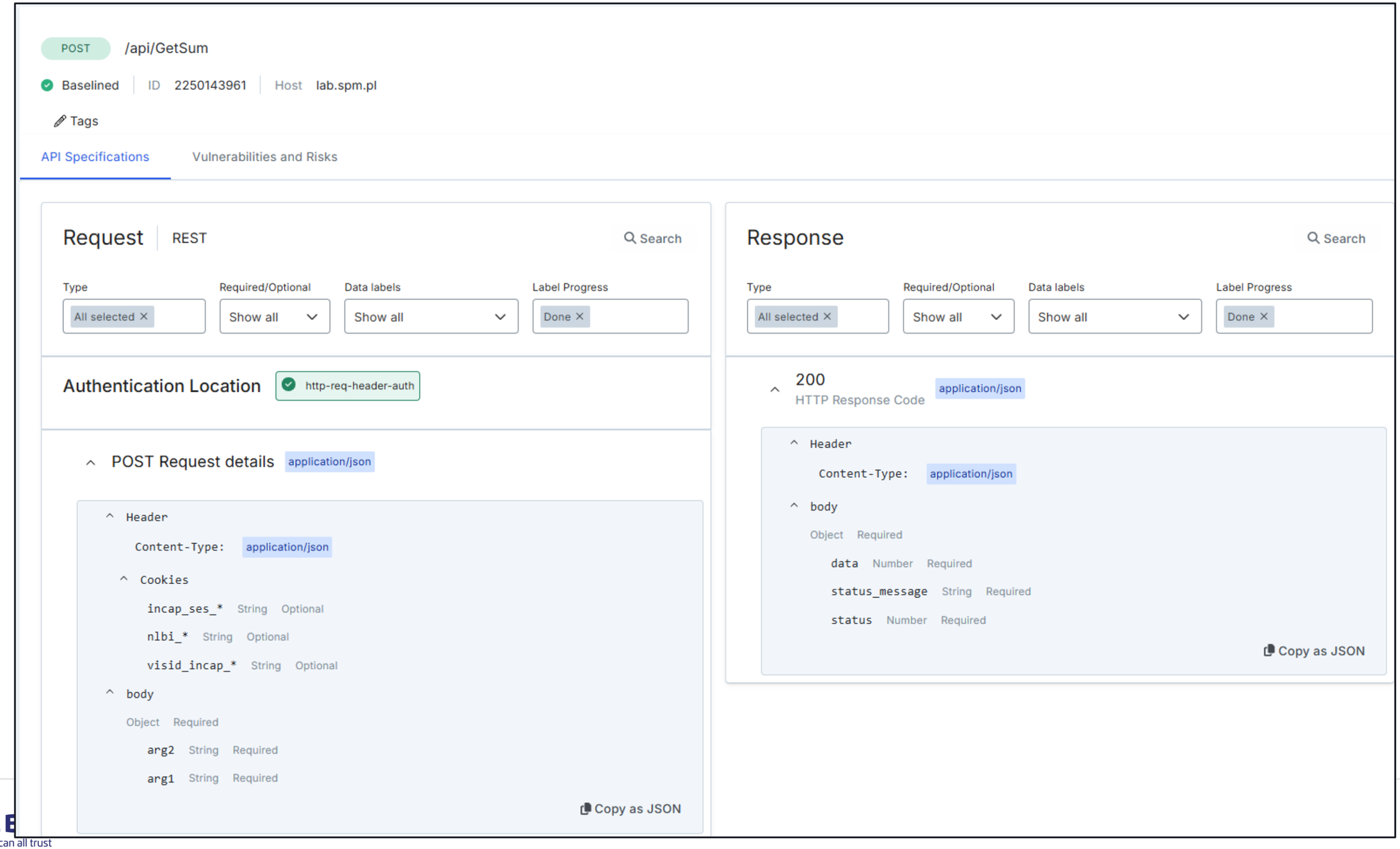
CyberSecurityPanel

demo

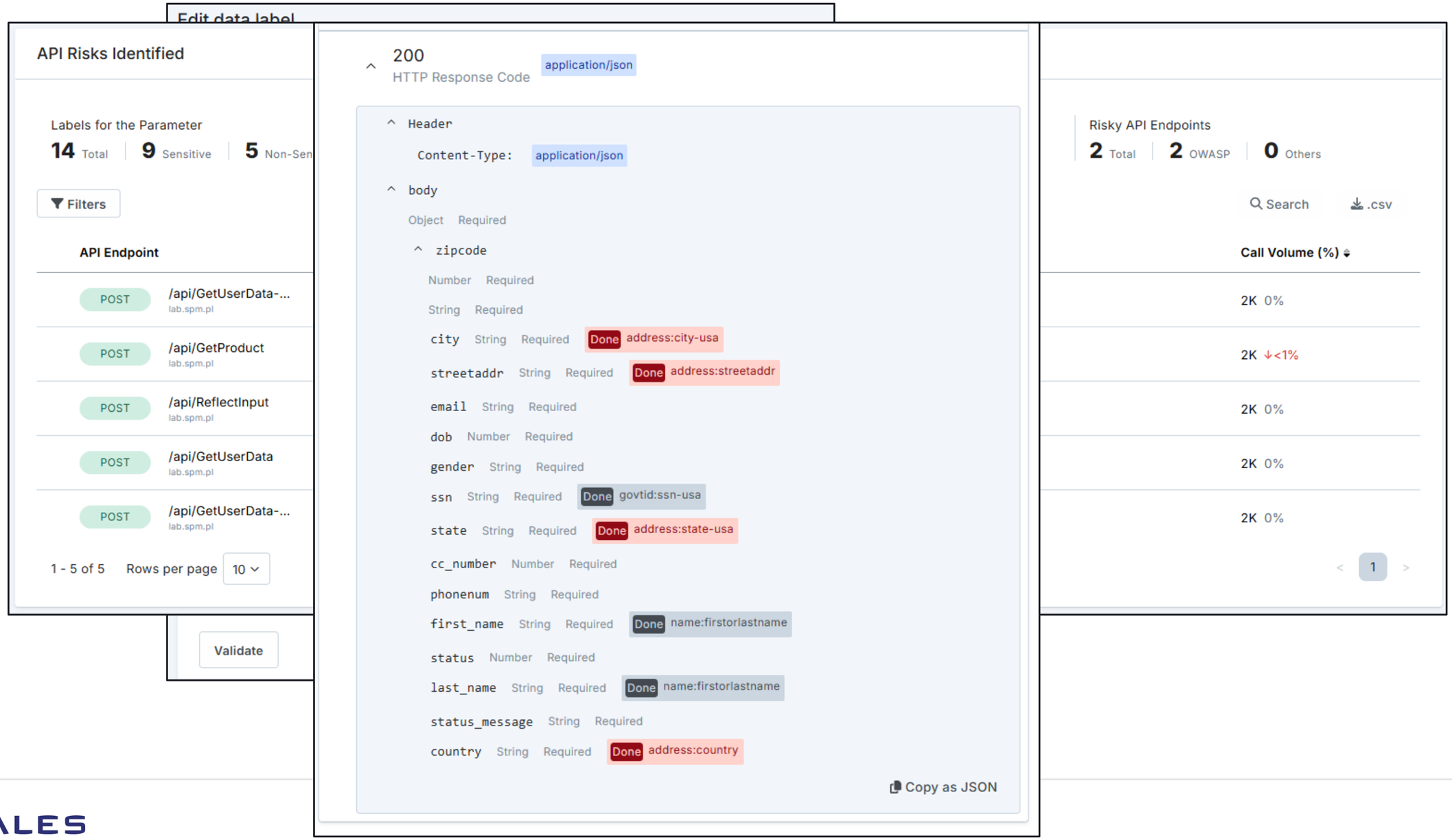
THALES
Building a future we can all trust

CLICO

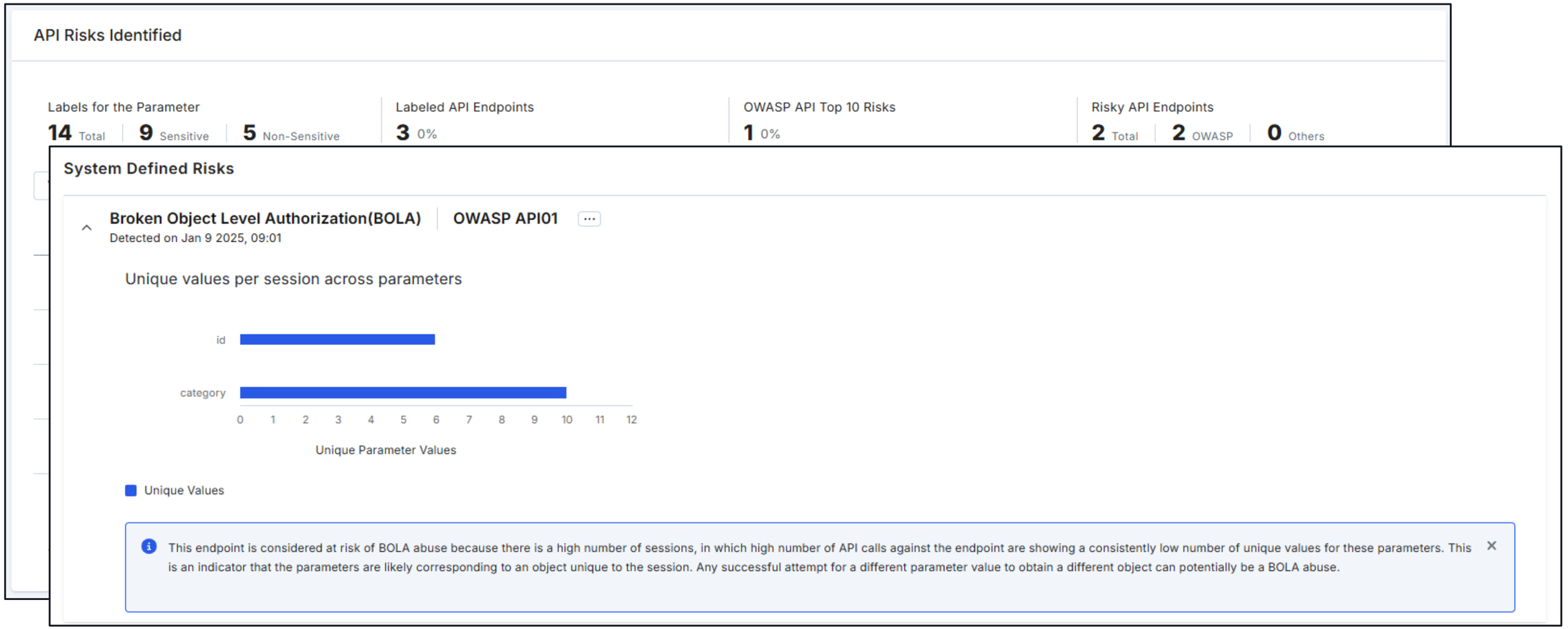
API Discovery



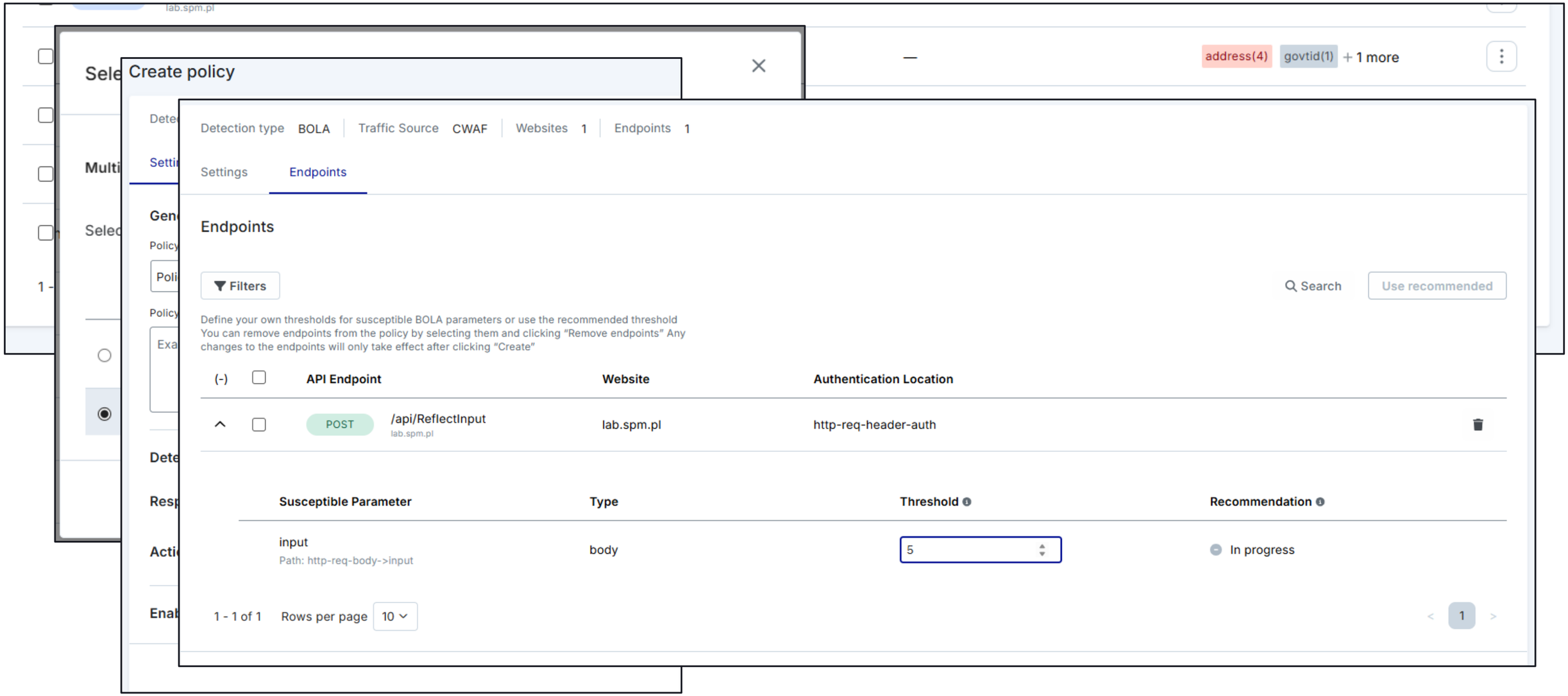
API Data Classification



API Threat Detection



API Threat Protection (new)



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

THALES
Building a future we can all trust

CLICO

CyberSecurityPanel