Session 1:
The new OWASP Top 10 API Security 2023

James Lee | Security Solutions Architect | F5

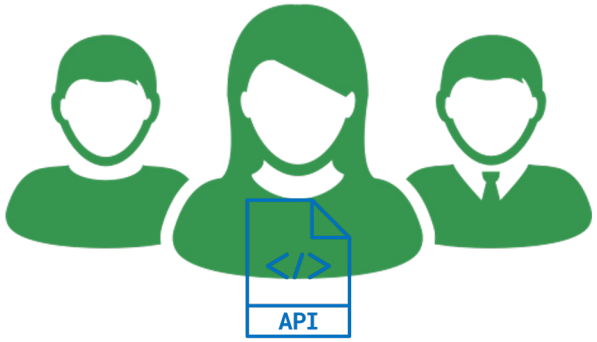
API Security is in the Grey area…

API Owners and SecOps

SecOps

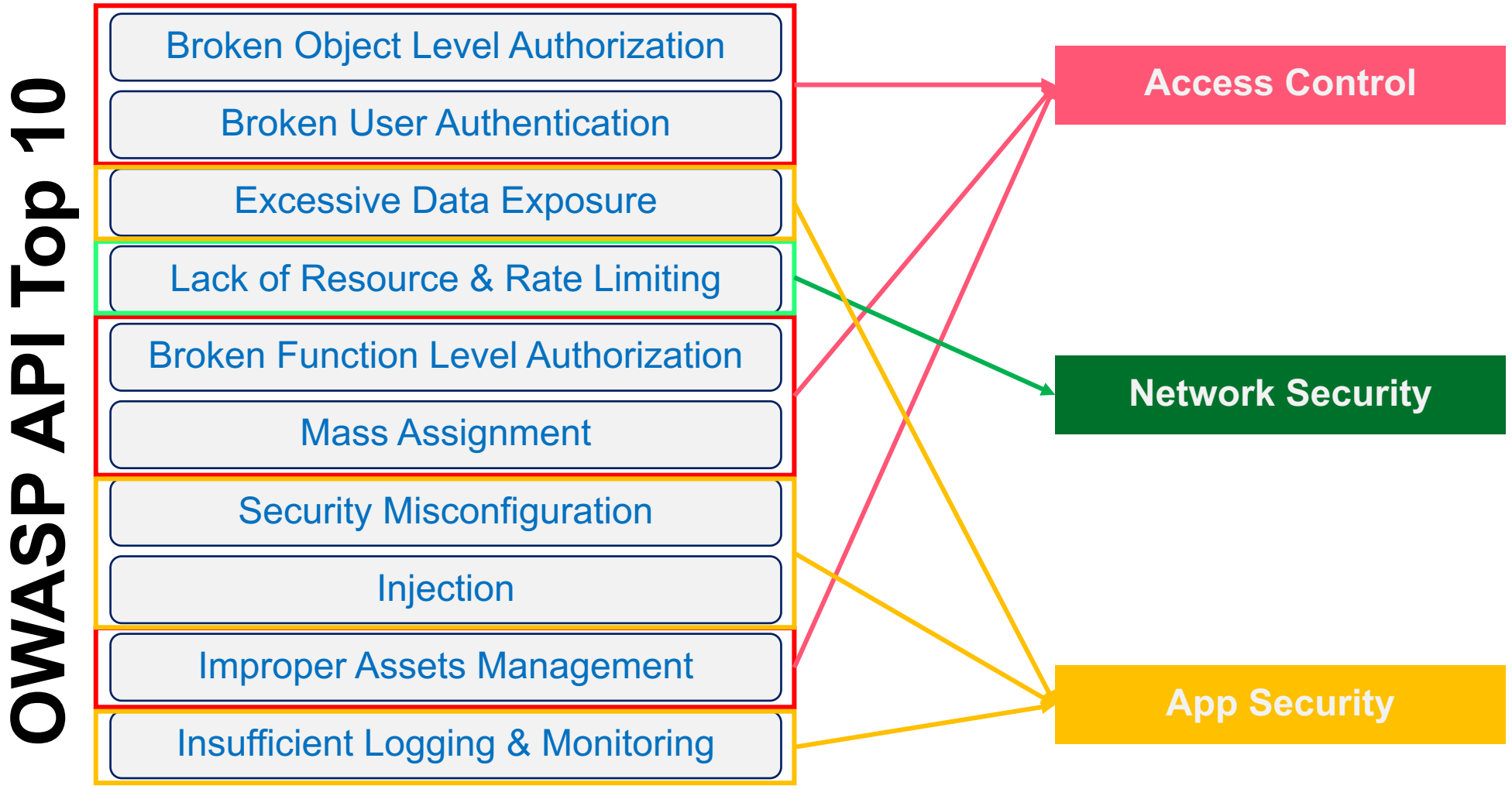
- Responsible for the organization’s security policy and compliance
- Operating security policies in NG F/W, WAF, and Anti-Virus
- Managing all alerts, including FPs and FNs
- Normally, SecOps provides the guideline of API security to API owners but they don't have control and visibility of API G/W or API management.

API Owners (Dev Team)



- Responsible for designing the business logic with APIs
- Operating API G/W, API management and required IAM(Identity and Access Management) for APIs
- API Owners are not considered security experts.
- Normally, API Owners manage the API G/W policies and code-level security but do not manage the organization-wide security policy or compliance.

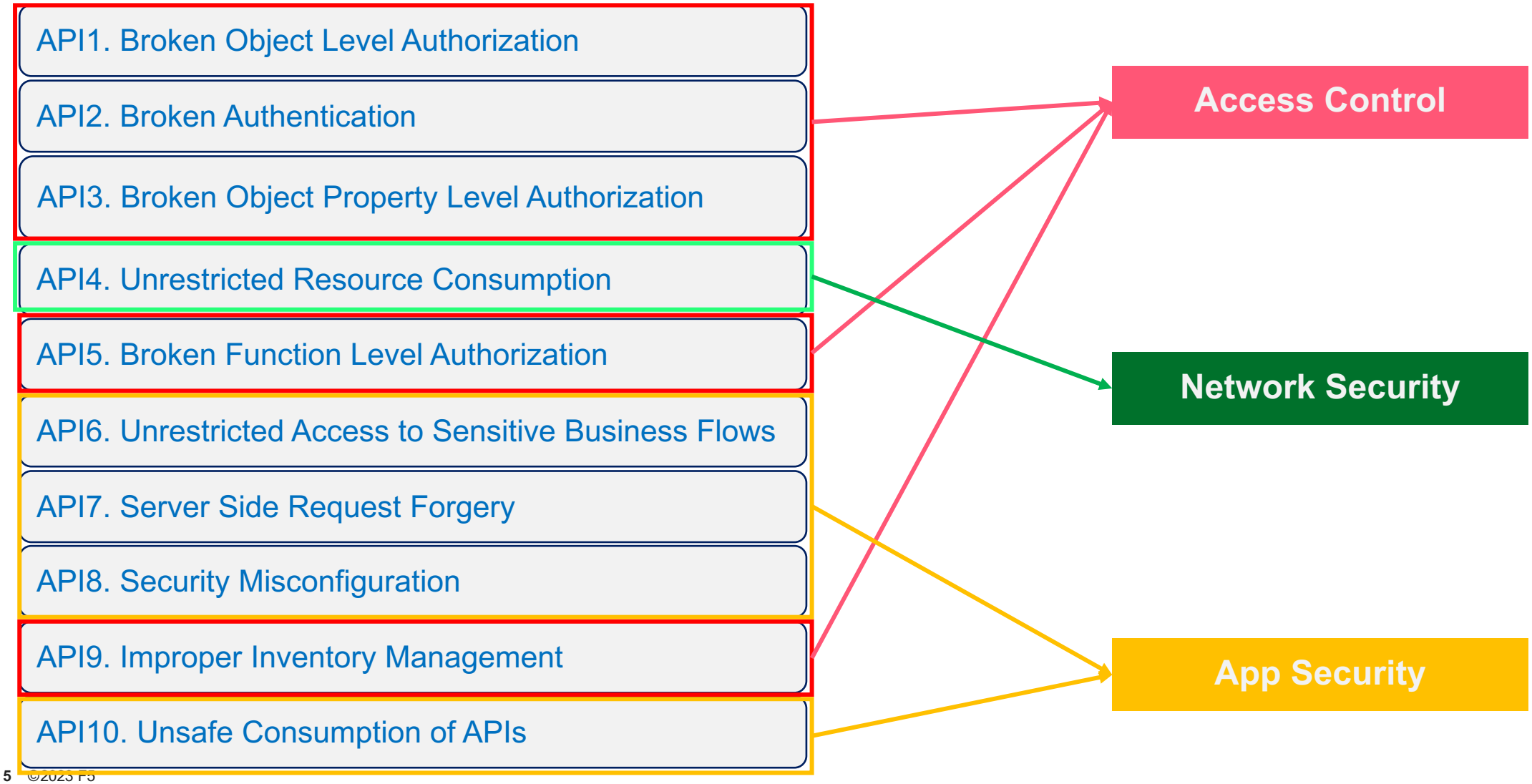
OWASP API Top 10 - 2019



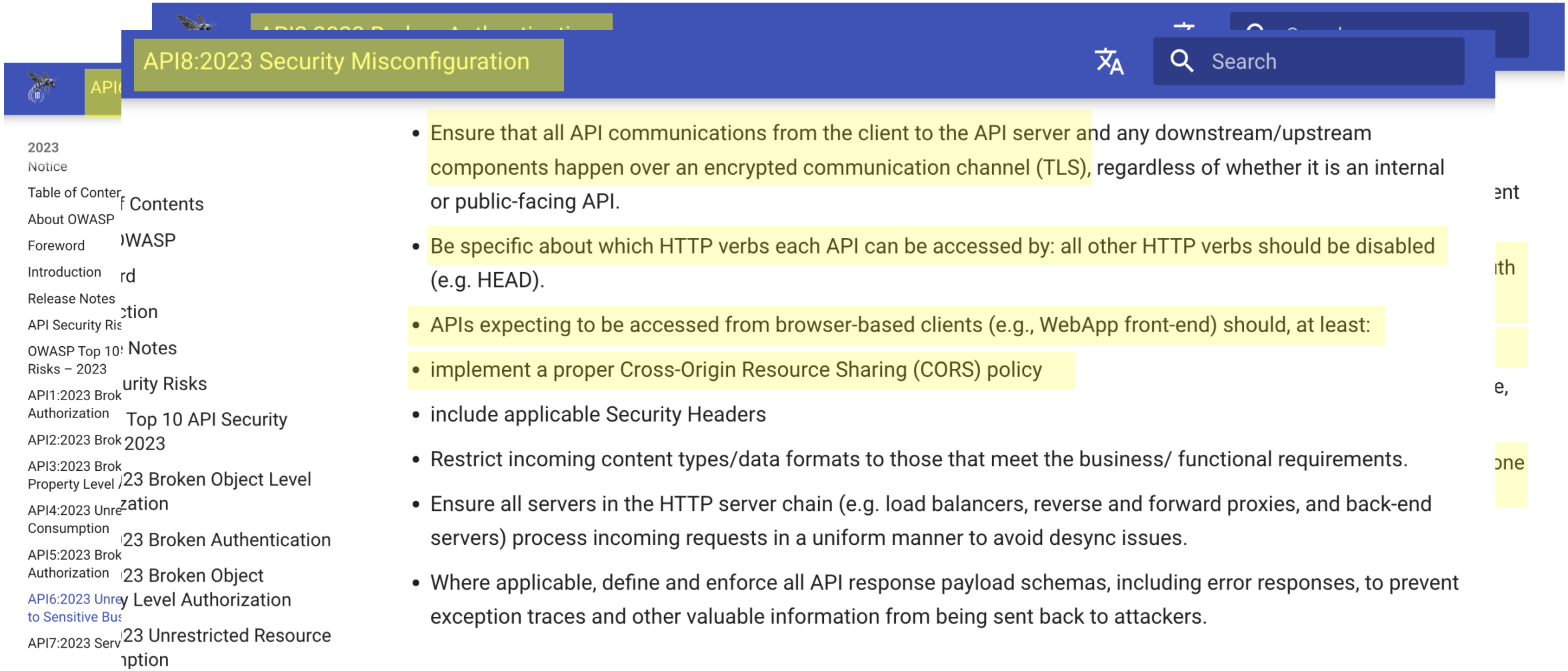
OWASP API Top 10 - 2023

Broken Object Level Authorization	Broken Object Level Authorization
Broken User Authentication	Broken Authentication
Excessive Data Exposure	Broken Object Property Level Authorization
Lack of Resource & Rate Limiting	Unrestricted Resource Consumption
Broken Function Level Authorization	Broken Function Level Authorization
Mass Assignment	Unrestricted Access to Sensitive Business Flows
Security Misconfiguration	Server Side Request Forgery
Injection	Security Misconfiguration
Improper Assets Management	Improper Inventory Management
Insufficient Logging & Monitoring	Unsafe Consumption of APIs

OWASP API Top 10 - 2023



How Can You Comply with the new OWASP 2023?

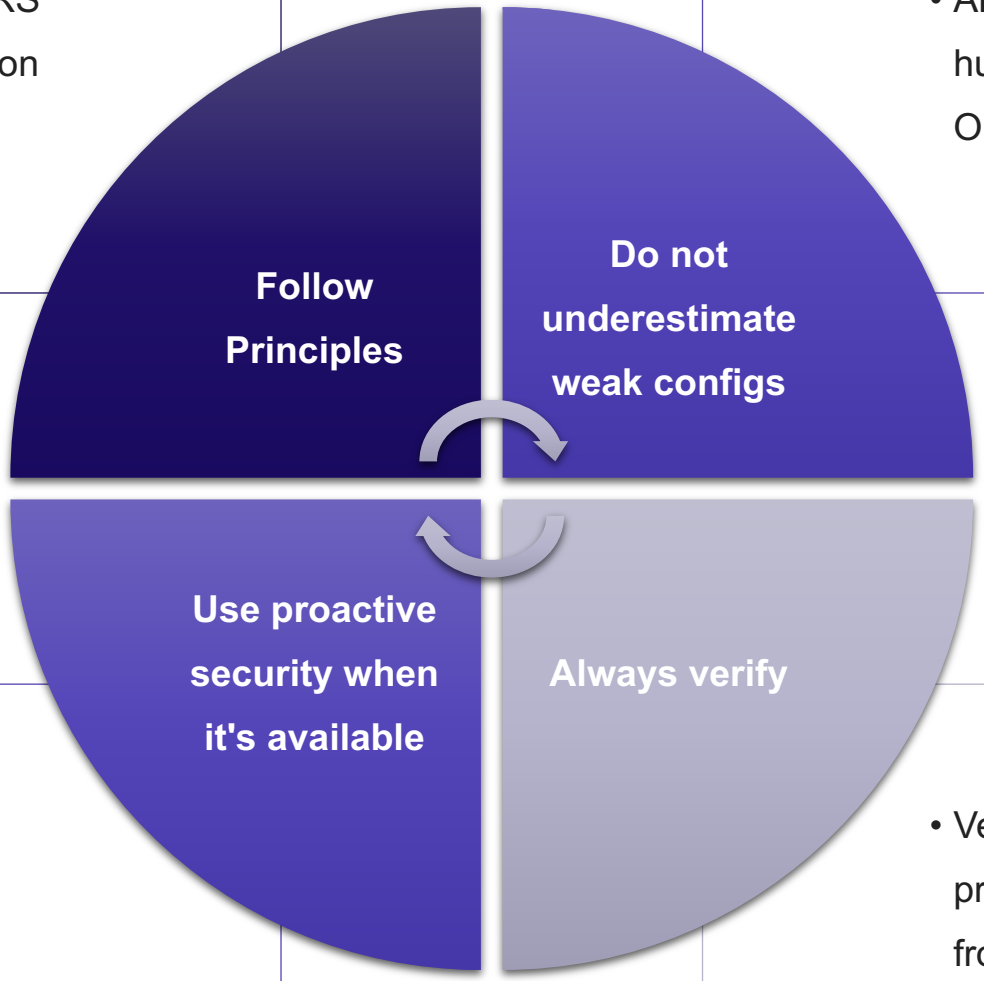


- Ensure that all API communications from the client to the API server and any downstream/upstream components happen over an encrypted communication channel (TLS), regardless of whether it is an internal or public-facing API.
- Be specific about which HTTP verbs each API can be accessed by: all other HTTP verbs should be disabled (e.g. HEAD).
- APIs expecting to be accessed from browser-based clients (e.g., WebApp front-end) should, at least:
- implement a proper Cross-Origin Resource Sharing (CORS) policy
- include applicable Security Headers
- Restrict incoming content types/data formats to those that meet the business/ functional requirements.
- Ensure all servers in the HTTP server chain (e.g. load balancers, reverse and forward proxies, and back-end servers) process incoming requests in a uniform manner to avoid desync issues.
- Where applicable, define and enforce all API response payload schemas, including error responses, to prevent exception traces and other valuable information from being sent back to attackers.

Guidelines of the new OWASP API Top 10 - 2023

- Blocking ToR IPs, CORS configuration, redirection handlings and etc

- API key, Human/Non-human detection and OpenAPI validation

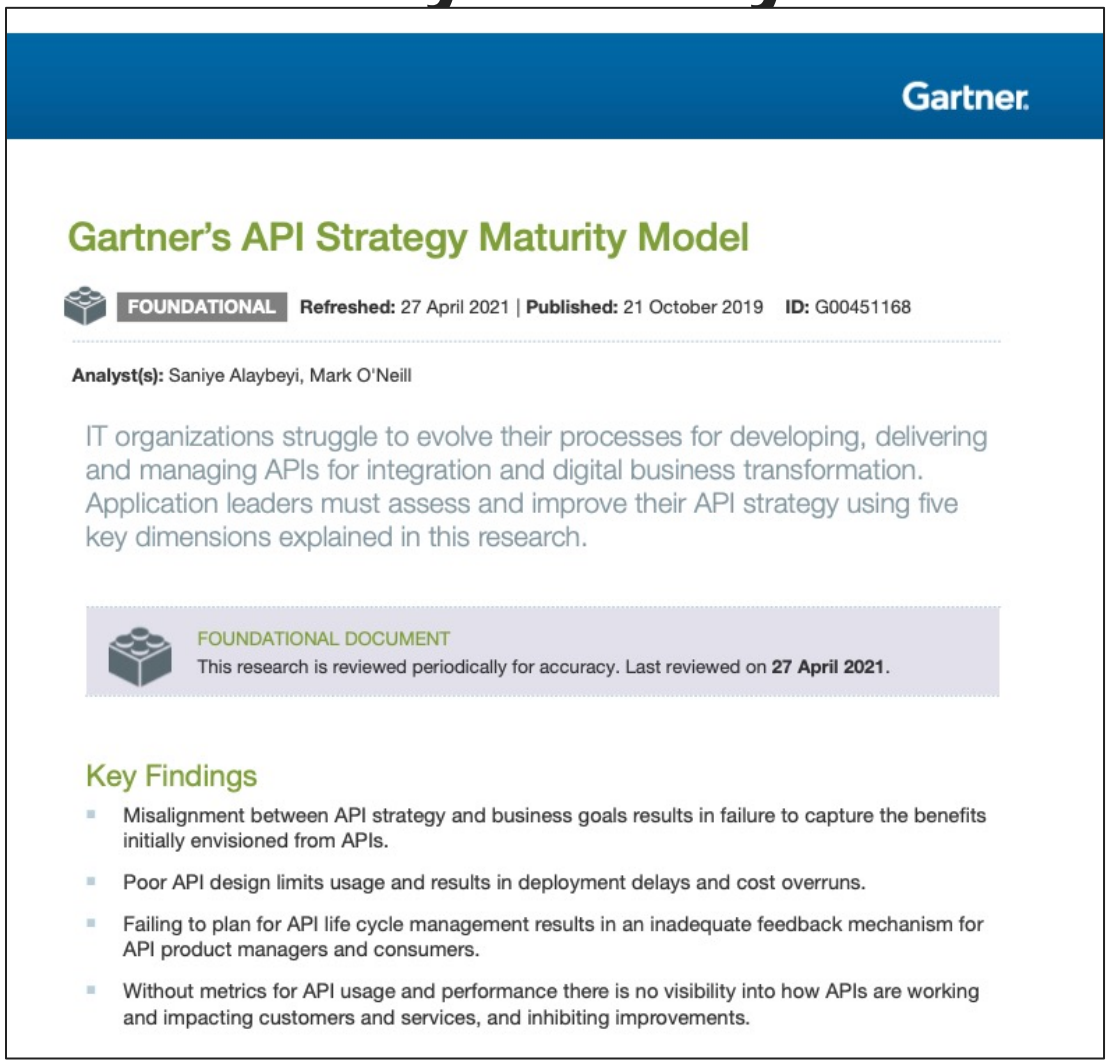


- Rate limiting, Bot detection, SSRF detection and etc.

- Verify the data and privilege. Even the one from the trusted partners

API Security Maturity Model

API Security Maturity Model



Gartner

Gartner's API Strategy Maturity Model

FOUNDATIONAL **Refreshed:** 27 April 2021 | **Published:** 21 October 2019 **ID:** G00451168

Analyst(s): Saniye Alaybeyi, Mark O'Neill

IT organizations struggle to evolve their processes for developing, delivering and managing APIs for integration and digital business transformation. Application leaders must assess and improve their API strategy using five key dimensions explained in this research.

FOUNDATIONAL DOCUMENT

This research is reviewed periodically for accuracy. Last reviewed on **27 April 2021**.

Key Findings

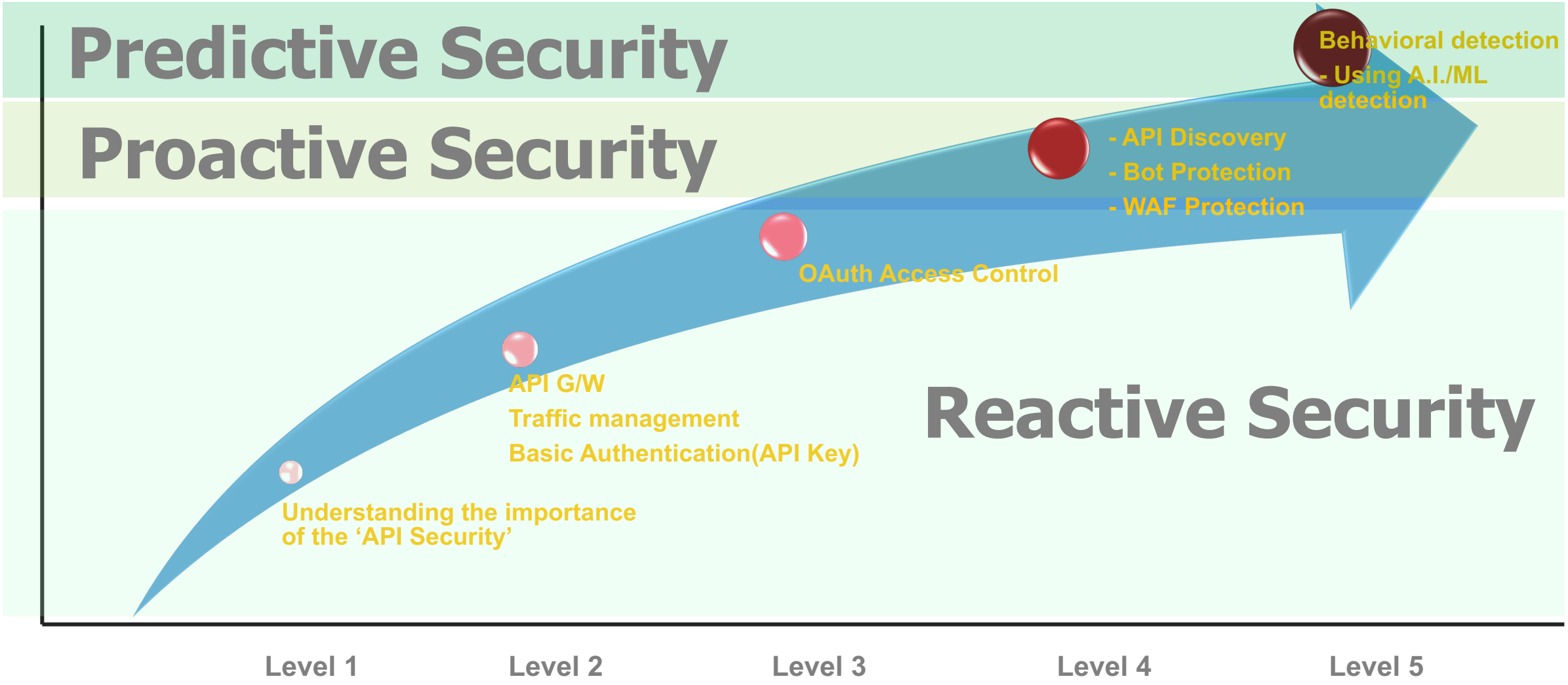
- Misalignment between API strategy and business goals results in failure to capture the benefits initially envisioned from APIs.
- Poor API design limits usage and results in deployment delays and cost overruns.
- Failing to plan for API life cycle management results in an inadequate feedback mechanism for API product managers and consumers.
- Without metrics for API usage and performance there is no visibility into how APIs are working and impacting customers and services, and inhibiting improvements.

API Security Maturity Model

Communications

- **Level 0:** There is no evidence or awareness for authentication, traffic management and privacy mechanisms or policies.
- **Level 1:** API teams understand the importance of an authentication, authorization, traffic management, quality of service (caching), interface protocols and security mechanisms and policies. Some of these basic mechanisms exist in a few silos within the organization and are implemented at various levels, but without API gateway in place.
- **Level 2:** Teams are implementing authentication, authorization and messaging mechanisms based on industry-accepted practices but in isolation of each other. Beyond individual team or solution implementation, standards and policies do not exist for security, interface, data privacy and traffic management (throttling). Instead, these are being implemented on a team-by-team or solution-by-solution basis, typically using API gateways.
- **Level 3:** Core principles, basic standards and policies are defined across the enterprise for API security, traffic management, quality of service, data privacy, service routing and orchestration. Standards are adopted at various levels across the organization, in many cases reactively.
- **Level 4:** API security, privacy, quality and communication standards are being proactively and consistently adopted across the organization. There is active monitoring and proactive intervention ensuring compliance with the defined standards. Developer consumption quotas and traffic prioritization mechanisms are in place. Robust caching, service orchestration and routing (load balancing) capabilities are in place. These may be provided by a full life cycle API management solution, which includes API gateways.
- **Level 5:** Organizations are transitioning their security and traffic management capabilities from proactive to predictive. Full life cycle API management and web application firewalls (WAFs) work together in unison to detect threats and anticipate traffic and required service levels. Network behavior analysis and content inspection mechanisms are in place to detect misuse and attacks. Visualization and advanced reporting monitoring mechanisms are in place.

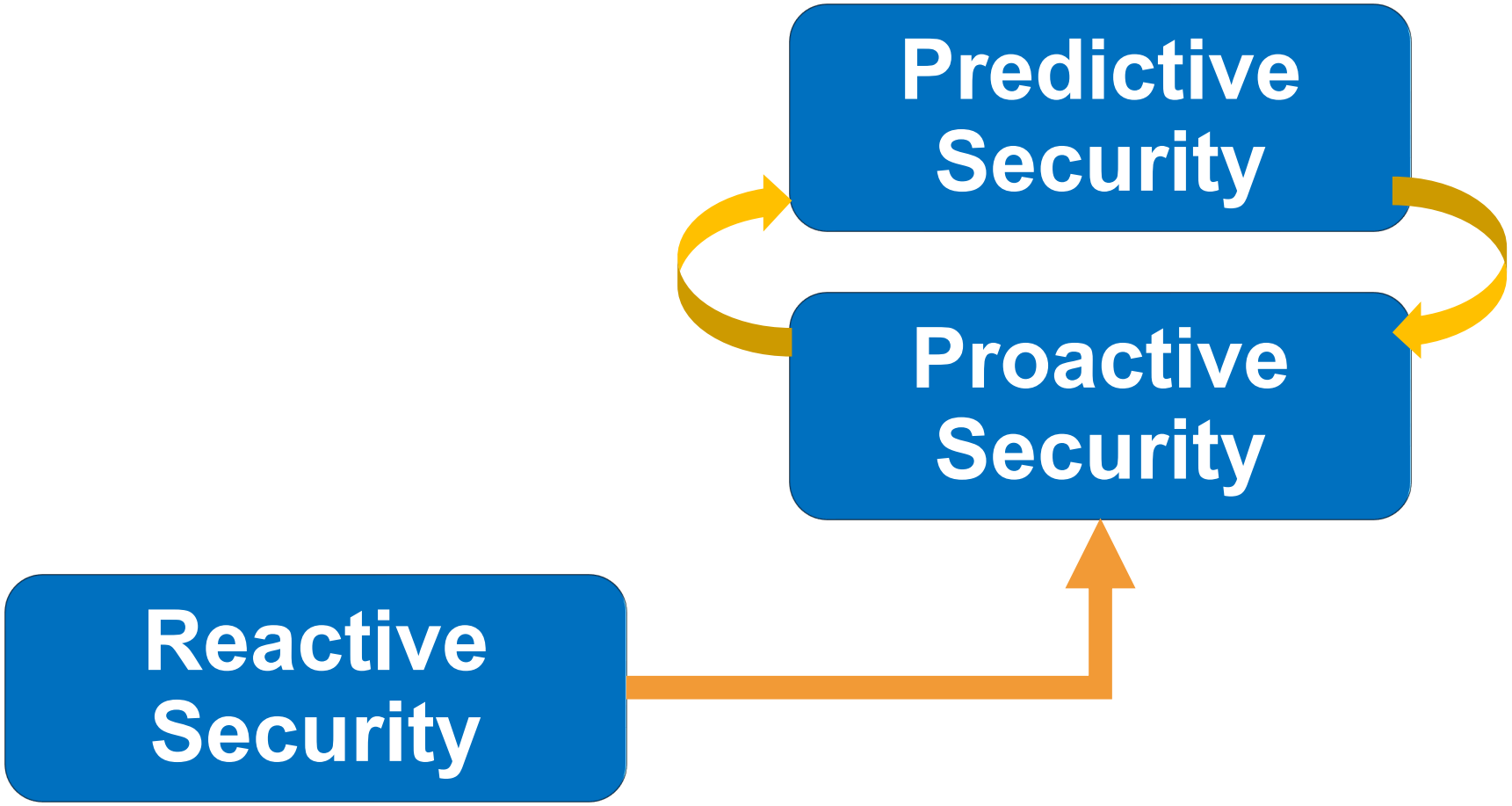
API Security Maturity Model



Reactive vs Proactive vs Predictive

	Reactive	Proactive	Predictive
Premise	• Reactive cyber security introduces defending against attacks that have already happened.	• Proactive cyber security involves identifying and addressing security risks before an attack occurs.	• Using contextual analysis to identify threats before they become incidents
Examples	• Adding IPs to Deny-list after 10 failed logon attempts	• Payload inspection by a WAF/WAAP	• Behavioral – Malicious or Suspicious user and API client detection
Benefits	• Stop-gap when all controls fail	• Real runtime protection	• Advance warning and protection
Powered By	• Logs and telemetry data indicating attacks	• Real time analysis of traffic	• Machine learning on telemetry data
Gartner Maturity Model Levels	• Level 1 ~ Level 3	• Level 4	• Level 5

API Security Maturity Model at Runtime



Continuous Improvement

Using AI/ML to reduce False Positive and improve accuracy

Session 1: The new OWASP Top 10 API Security 2023

API8:2023 – Security Misconfiguration

API8: Security Misconfiguration

Security misconfiguration vulnerabilities occur when an API component is susceptible to attack due to a misconfiguration or nonsecure configuration

- API Inventory/ documentation incomplete or missing
- APIs don’t conform to OpenAPI specifications
- Authentication token includes insecure configuration
- PII data exists in the JWT or API request body
- CORS misconfiguration

- An F&B outlet had an API not meant for external users exposed putting PII information for 100 Million users at risk
- Attacker used brute force to detect the vulnerable API endpoint

Source: samcurry.net

API8: Security Misconfiguration

Examples

Case#01.

- An attacker finds the .bash_history file under the root directory of the server, which contains commands used by the DevOps team to access the API.

Case#02.

- An attacker finds some API endpoints that are accessible from the internet but don’t have the authentication mechanism.

Case#03.

- Some API endpoints include potentially critical data in their token or header or body, and an attacker can exploit those data potentially.

API8: Security Misconfiguration

How to Stop the Security Misconfiguration

API8:2023 Security Misconfiguration

Search

f Contents
)WASP
rd
ction
Notes
urity Risks
Top 10 API Security 2023
023 Broken Object Level zation
023 Broken Authentication
023 Broken Object y Level Authorization
023 Unrestricted Resource nption

- Ensure that all API communications from the client to the API server and any downstream/upstream components happen over an encrypted communication channel (TLS), regardless of whether it is an internal or public-facing API.
- Be specific about which HTTP verbs each API can be accessed by: all other HTTP verbs should be disabled (e.g. HEAD).
- APIs expecting to be accessed from browser-based clients (e.g., WebApp front-end) should, at least:
- implement a proper Cross-Origin Resource Sharing (CORS) policy
- include applicable Security Headers
- Restrict incoming content types/data formats to those that meet the business/ functional requirements.
- Ensure all servers in the HTTP server chain (e.g. load balancers, reverse and forward proxies, and back-end servers) process incoming requests in a uniform manner to avoid desync issues.
- Where applicable, define and enforce all API response payload schemas, including error responses, to prevent exception traces and other valuable information from being sent back to attackers.

API8: Security Misconfiguration

How to Stop the Security Misconfiguration

OpenAPI Spec Validation

- It is always recommended to enable OpenAPI validation to perform a basic schema validation for your APIs.

Strong AuthN/AuthZ

- Need to consider strong authN/authZ methods for your APIs with standard-based protocols such as OAuth/OIDC or SAML.

Continuous Monitoring

- Continuous assessment of the security effectiveness of API endpoints.

HTTP Methods Control

- Enable only relevant HTTP methods for the specific API endpoints.

TLS Policy Setting

- Need to have the capability to set up the different TLS policies based on the backend apps' requirements.

CORS Policy

- Need to configure the correct CORS settings in your apps.

API8: Security Misconfiguration

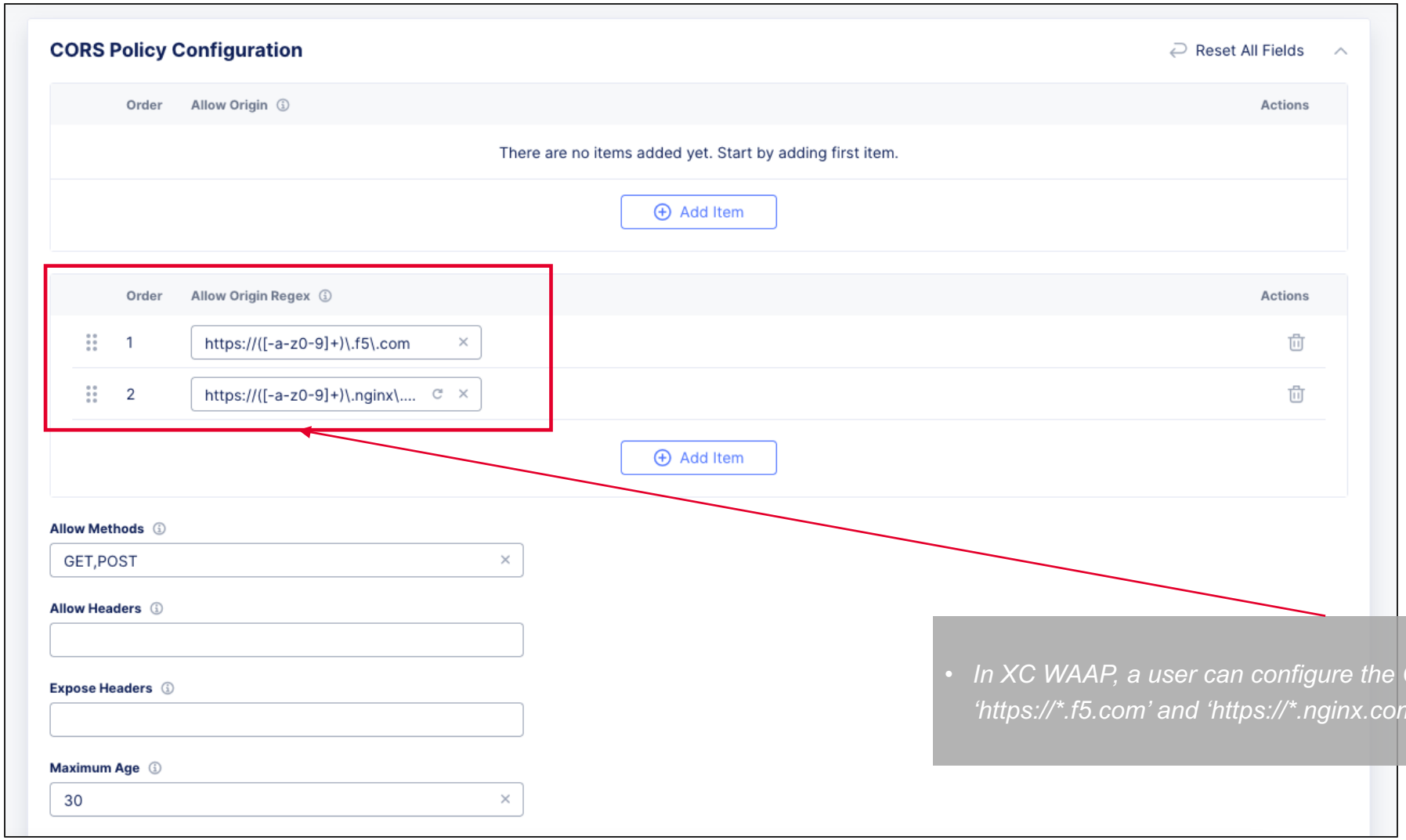
Protection#01 – CORS Configuration

```
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ % curl -I -H "Origin: https://www.evil.com"  https://james.apac-ent.f5demos.com
HTTP/2 200
access-control-allow-origin: *
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
feature-policy: payment 'self'
x-recruiting: /#/jobs
accept-ranges: bytes
cache-control: public, max-age=0
last-modified: Thu, 20 Jul 2023 02:55:19 GMT
etag: W/"7c3-18971392a77"
content-type: text/html; charset=UTF-8
content-length: 1987
vary: Accept-Encoding
date: Thu, 20 Jul 2023 12:24:33 GMT
x-envoy-upstream-service-time: 7
set-cookie: 8bd85=1689855873089-779805097; Path=/; Domain=f5demos.com; Expires=Fri, 20 Oct 2023 12:24:33 GMT
set-cookie: 8bd803=HHC33JWEDc6kXUZPt2cFmIril9/iBbs+FTi7yazFl6ENk9KHP+WVa6md5t+aoW2N1c9+HdZu0Ciauth989Gjxhn1F4wGZnfnfxak12ZPLkCiVTOXPNDxbUgj5
AgQ9QOHmluTsPTbNyFS8tau3dc6McRprJKtNS4wDTSzuLNXD9PDiu6B; path=/
x-volterra-location: sg3-sin
server: volt-adc
```

- *The webserver has a vulnerable CORS setting with the wildcard(*).*

API8: Security Misconfiguration

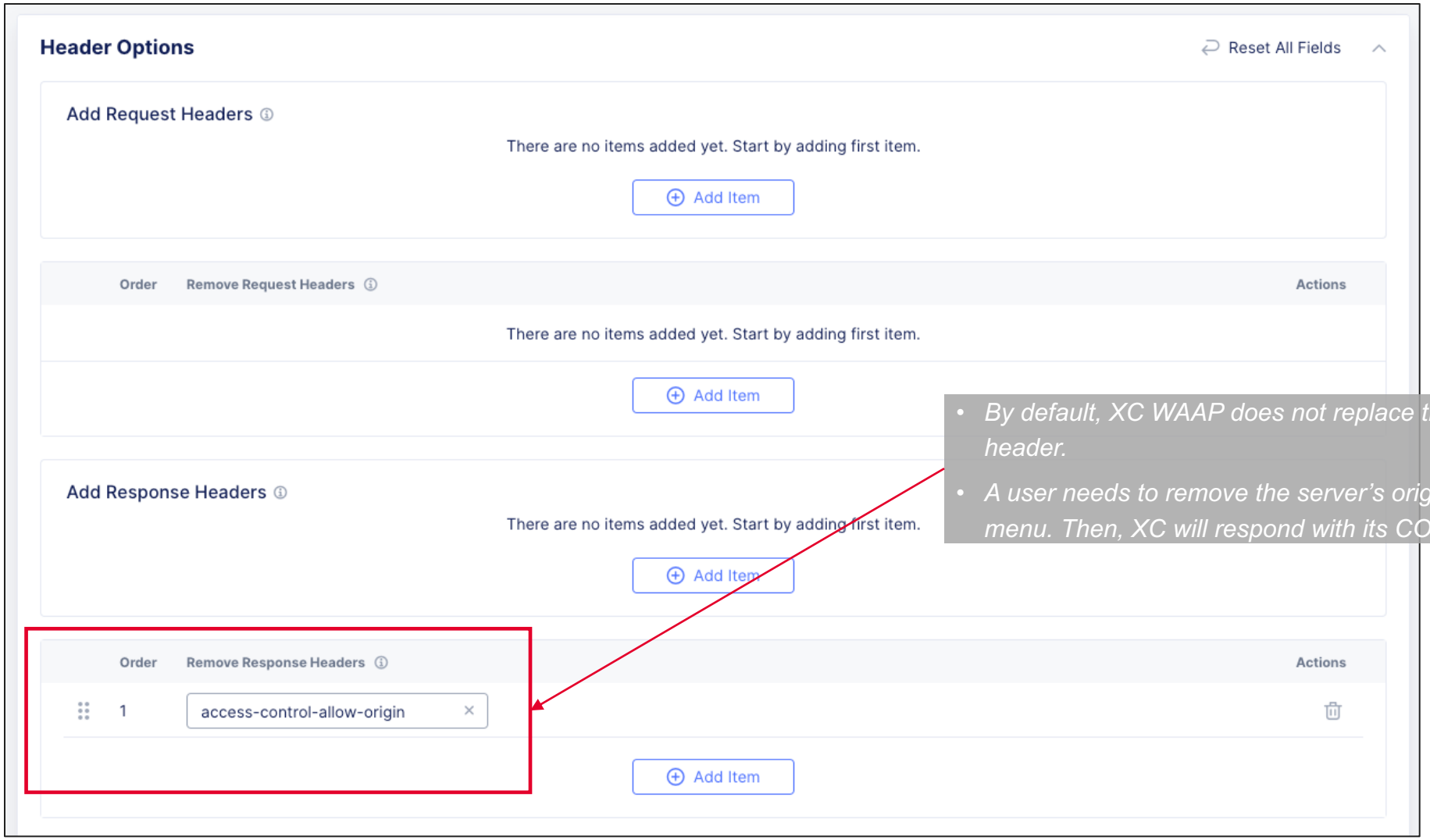
Protection#01 – CORS Configuration



- *In XC WAAP, a user can configure the CORS policy. In this case, 'https://*.f5.com' and 'https://*.nginx.com' are only allowed.*

API8: Security Misconfiguration

Protection#01 – CORS Configuration



- *By default, XC WAAP does not replace the server's original header.*
- *A user needs to remove the server's original response using this menu. Then, XC will respond with its CORS configuration.*

API8: Security Misconfiguration

Protection#01 – CORS Configuration

```
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ % curl -I -H "Origin: https://www.f5.com"  https://james.apac-ent.f5demos.com
HTTP/2 200
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
feature-policy: payment 'self'
x-recruiting: /#/jobs
accept-ranges: bytes
cache-control: public, max-age=0
last-modified: Thu, 20 Jul 2023 02:55:19 GMT
etag: W/"7c3-18971392a77"
content-type: text/html; charset=UTF-8
content-length: 1987
vary: Accept-Encoding
date: Thu, 20 Jul 2023 12:17:25 GMT
x-envoy-upstream-service-time: 7
set-cookie: 8bd85=1689855445431-141961141; Path=/; Domain=f5demos.com; Expires=Fri, 20 Oct 2023 12:17:25 GMT
set-cookie: 8bd803=j7PKaEUHKf1anRb6AsNw/pcdOJzZVzEniY2/kU3Q0o+O5NnE3rqQ/f[illegible]QE+Pg5tfGzg7yxgFiUlWzh2pgOWCbkRLwE7knZ3HC3hN7C
u6gUgUvgpBGPT1cqsqVcpu1nSPAQBnrvx/hq0zbl7Diec079lTCmWC9; path=/
access-control-allow-origin: https://www.f5.com
x-volterra-location: sg3-sin
server: volt-adc
```

- *Since the request includes the correct origin server in the header, XC responds accordingly.*

API8: Security Misconfiguration

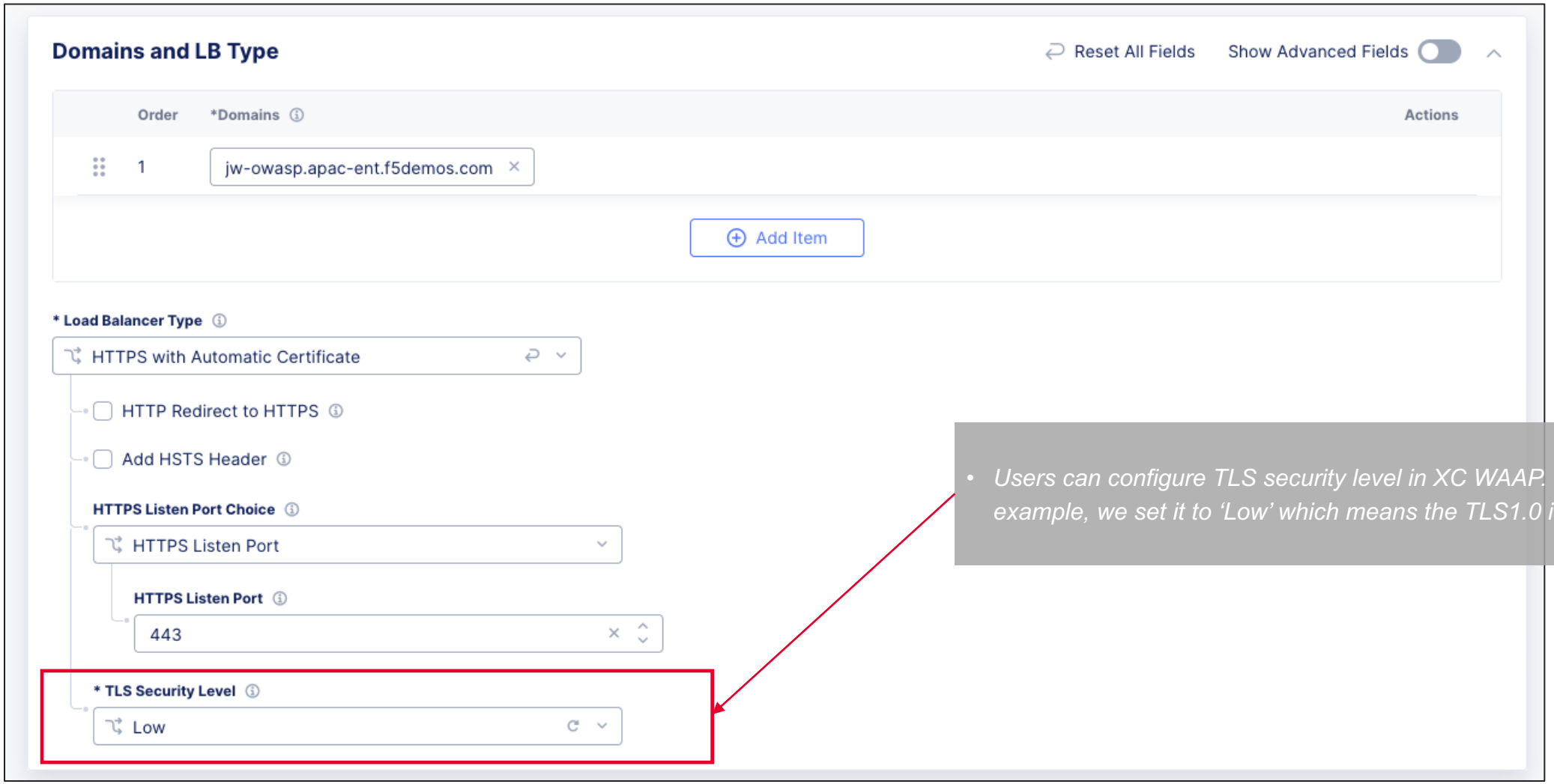
Protection#01 – CORS Configuration

```
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ % curl -I -H "Origin: https://www.evil.com"  https://james.apac-ent.f5demos.com
HTTP/2 200
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
feature-policy: payment 'self'
x-recruiting: /#/jobs
accept-ranges: bytes
cache-control: public, max-age=0
last-modified: Thu, 20 Jul 2023 02:55:19 GMT
etag: W/"7c3-18971392a77"
content-type: text/html; charset=UTF-8
content-length: 1987
vary: Accept-Encoding
date: Thu, 20 Jul 2023 12:21:03 GMT
x-envoy-upstream-service-time: 7
set-cookie: 8bd85=1689855663446-855534115; Path=/; Domain=f5demos.com; Expires=Fri, 20 Oct 2023 12:21:03 GMT
set-cookie: 8bd803=+IJPxIAYZFs3NCq1hi7XlZTxbirrP63rHRO2lO5yCXnyH9AbG5iuFM+9PKGWUc1Kl/G1rft51lWb1L5w3kI1p5wpxlx/OVQ86qIzhtR+m/8LxwHayEYx6Dq5S
zCspKaUbkx/mVdoXEs3Sjj4QsnWzmTG+gqBy34XgbZ6YDUaI0GHFRL1; path=/
x-volterra-location: sg3-sin
server: volt-adc
```

- *Since the request has the 'un-allowed' origin server, XC WAAP does not respond accordingly. Thus, the request would be rejected in the user's browser.*

API8: Security Misconfiguration

Protection#02 – TLS Configuration



- *Users can configure TLS security level in XC WAAP. In this example, we set it to 'Low' which means the TLS1.0 is allowed.*

API8: Security Misconfiguration

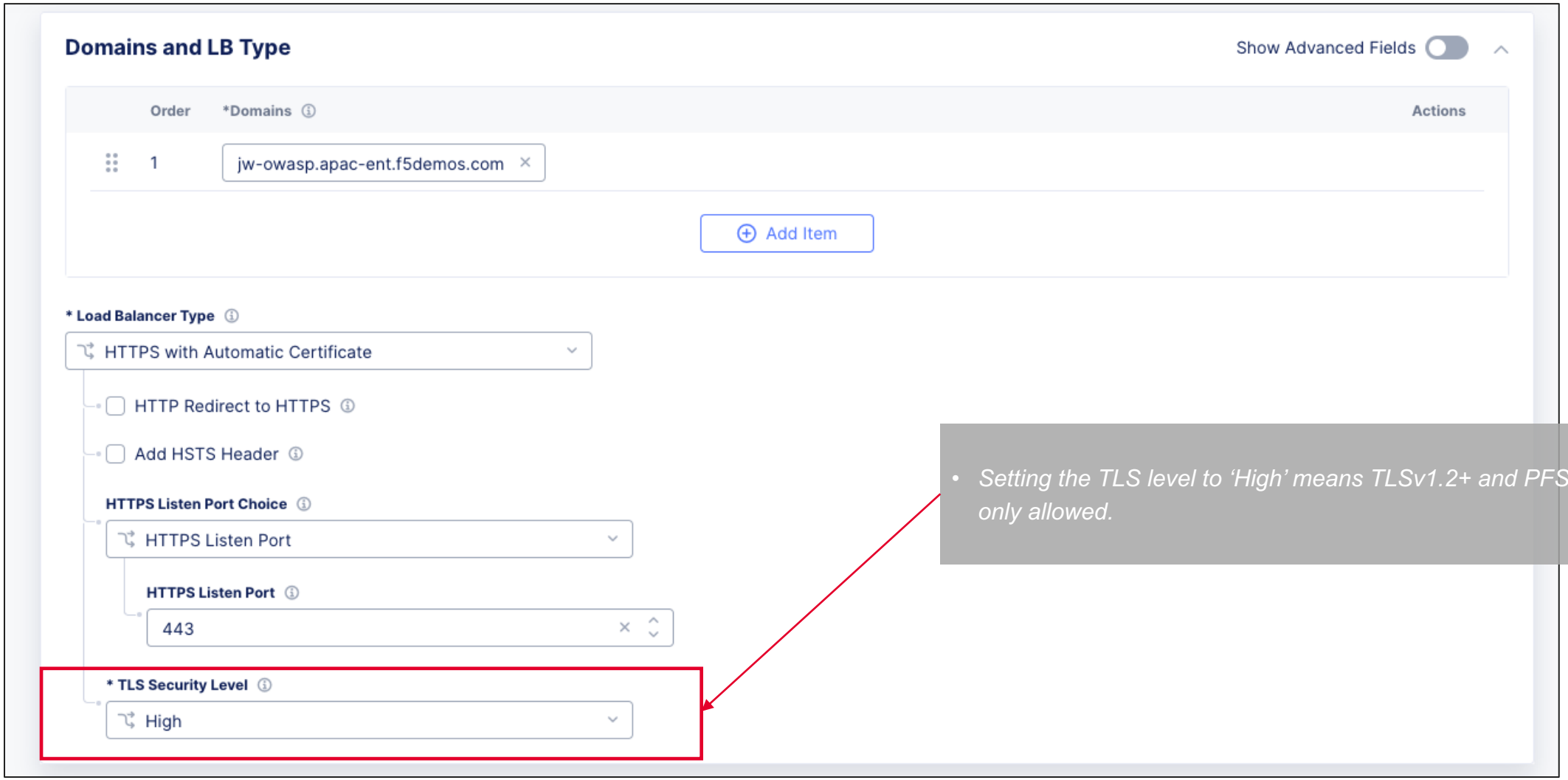
Protection#02 – TLS Configuration

```
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ % curl -vv --tlsv1.0  --tls-max 1.0 https://jw-owasp.apac-ent.f5demos.com/index1.php
*   Trying 72.19.3.189:443...
* Connected to jw-owasp.apac-ent.f5demos.com (72.19.3.189) port 443 (#0)
* ALPN: offers h2
* ALPN: offers http/1.1
*  CAfile: /etc/ssl/cert.pem
*  CApath: none
* [CONN-0-0][CF-SSL] TLSv1.0 (OUT), TLS handshake, Client hello (1):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS handshake, Server hello (2):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS handshake, Certificate (11):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS handshake, Server finished (14):
* [CONN-0-0][CF-SSL] TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* [CONN-0-0][CF-SSL] TLSv1.0 (OUT), TLS change cipher, Change cipher spec (1):
* [CONN-0-0][CF-SSL] TLSv1.0 (OUT), TLS handshake, Finished (20):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS change cipher, Change cipher spec (1):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1 / ECDHE-ECDSA-AES128-SHA
* ALPN: server accepted h2
* Server certificate:
*  subject: CN=jw-owasp.apac-ent.f5demos.com
*  start date: Jul 14 04:17:07 2023 GMT
*  expire date: Oct 12 04:17:06 2023 GMT
*  subjectAltName: host "jw-owasp.apac-ent.f5demos.com" matched cert's "jw-owasp.apac-ent.f5demos.com"
*  issuer: C=US; O=Let's Encrypt; CN=R3
*  SSL certificate verify ok.
```

- *TLS v1.0 works correctly.*

API8: Security Misconfiguration

Protection#02 – TLS Configuration



- *Setting the TLS level to 'High' means TLSv1.2+ and PFS will be only allowed.*

API8: Security Misconfiguration

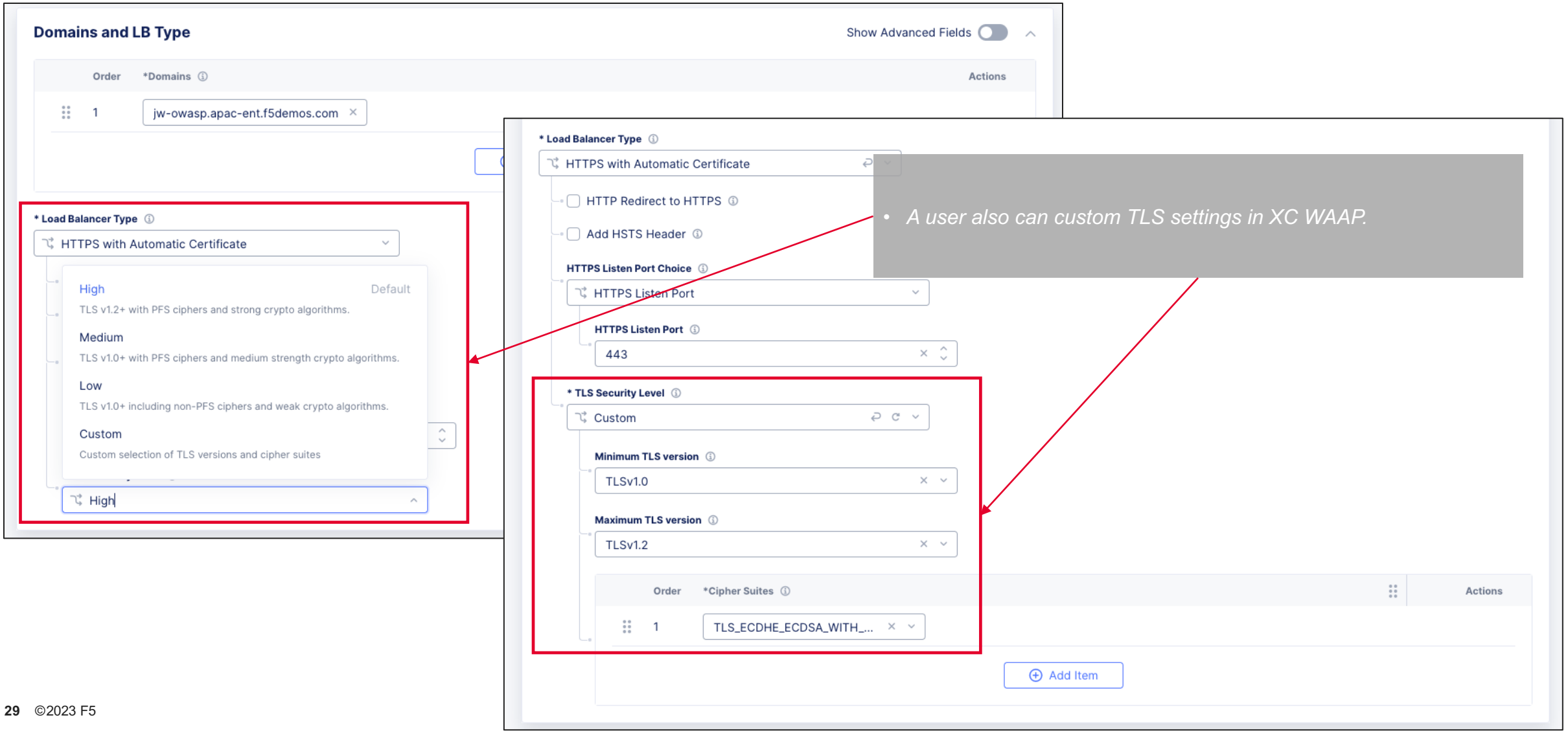
Protection#02 – TLS Configuration

```
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ % curl -vv --tlsv1.0  --tls-max 1.0 https://jw-owasp.apac-ent.f5demos.com/index1.php
*   Trying 72.19.3.189:443...
* Connected to jw-owasp.apac-ent.f5demos.com (72.19.3.189) port 443 (#0)
* ALPN: offers h2
* ALPN: offers http/1.1
*  CAfile: /etc/ssl/cert.pem
*  CApath: none
* [CONN-0-0][CF-SSL] TLSv1.0 (OUT), TLS handshake, Client hello (1):
* [CONN-0-0][CF-SSL] TLSv1.0 (IN), TLS alert, protocol version (582):
* LibreSSL/2.8.3: error:1400442E:SSL routines:CONNECT_CR_SRVR_HELLO:tlsv1 alert protocol version
* Closing connection 0
curl: (35) LibreSSL/2.8.3: error:1400442E:SSL routines:CONNECT_CR_SRVR_HELLO:tlsv1 alert protocol version
j.lee@C02FPFZPMD6M ~ %
j.lee@C02FPFZPMD6M ~ %
```

- *TLS v1.0 is NOT allowed in XC WAAP.*

API8: Security Misconfiguration

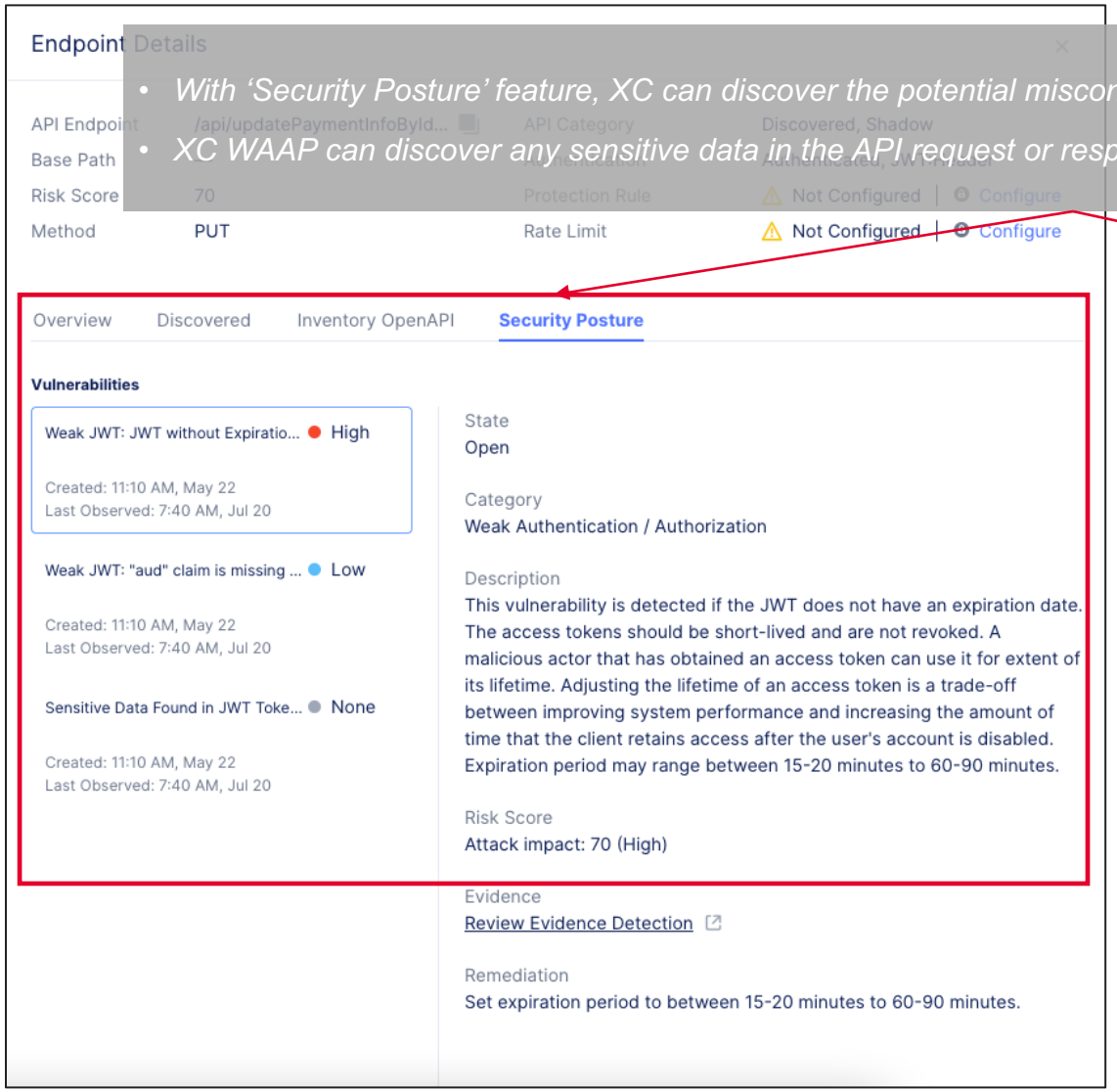
Protection#02 – TLS Configuration

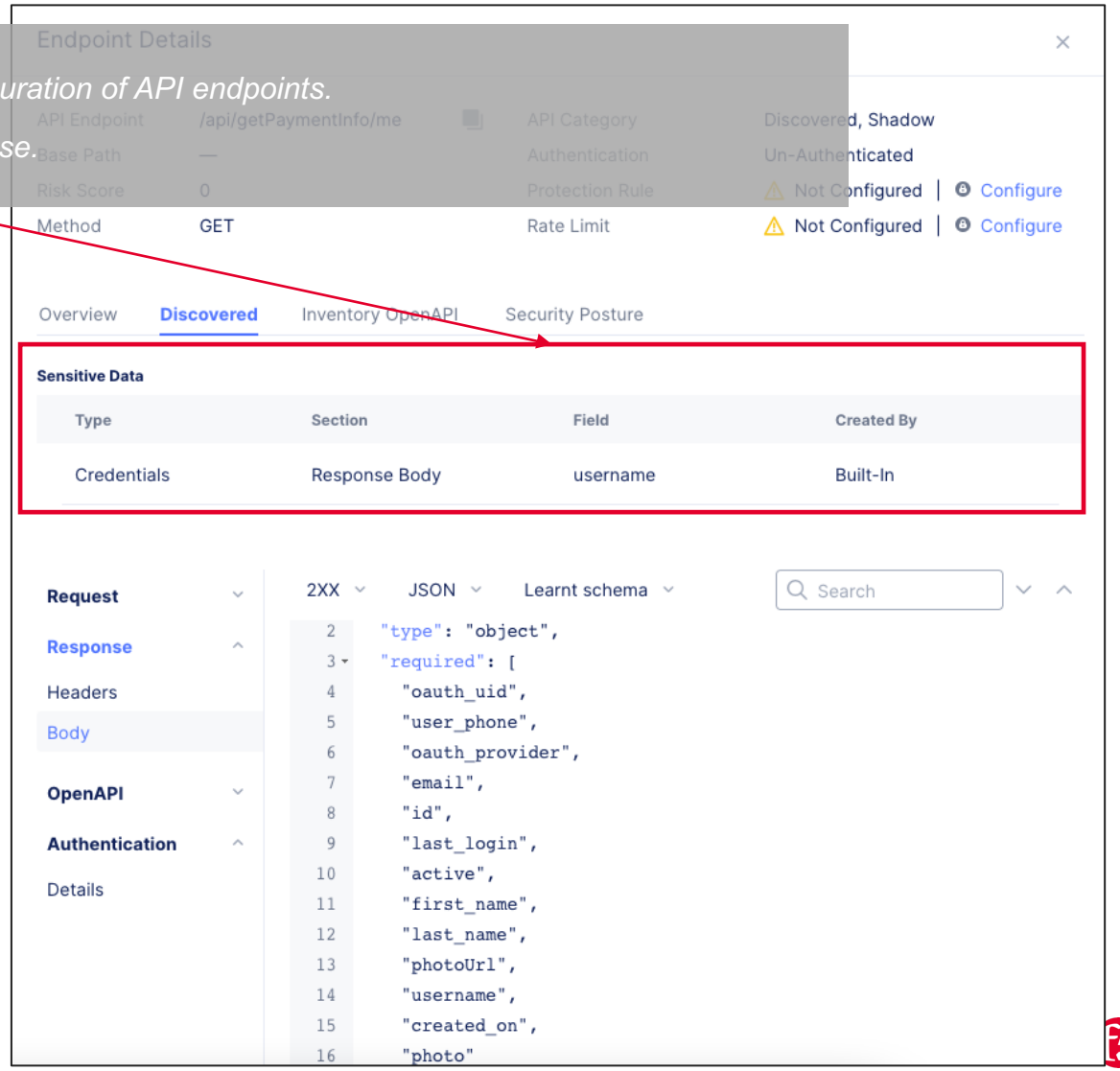


API8: Security Misconfiguration

Protection#03 – Continuous Monitoring

- *With 'Security Posture' feature, XC can discover the potential misconfiguration of API endpoints.*
- *XC WAAP can discover any sensitive data in the API request or response.*





API8: Security Misconfiguration

Protection#03 – Continuous Monitoring

API Endpoint	Method	Sensitive Data	Threat Level	Authentication S...	Authentication T...	API Category	Risk Score
/api/auth/oidc	GET	—	None	Authenticated	JWT:Header	Discovered Shadow	70
/api/getPaymentInfo/me	GET	Credentials	None	Un-Authenticated	—	Discovered Shadow	0
/trading/login.php	GET	—	High	Un-Authenticated	—	Discovered Shadow	0
/api/getStock	GET	—	None	Un-Authenticated	—	Discovered Shadow	0
[illegible]	GET	—	None	Un-Authenticated	—	Discovered Shadow	0

- *XC WAAP can discover the Shadow APIs, API authentication types, and their authentication state.*
- *In this example, some APIs don't have a proper auth mechanism(Un-Authenticated) and they're Shadow APIs which means API is unknown to API owners.*
- *Those APIs can be categorized as high-risky APIs potentially, and likely it happened because of 'Security Misconfiguration'.*

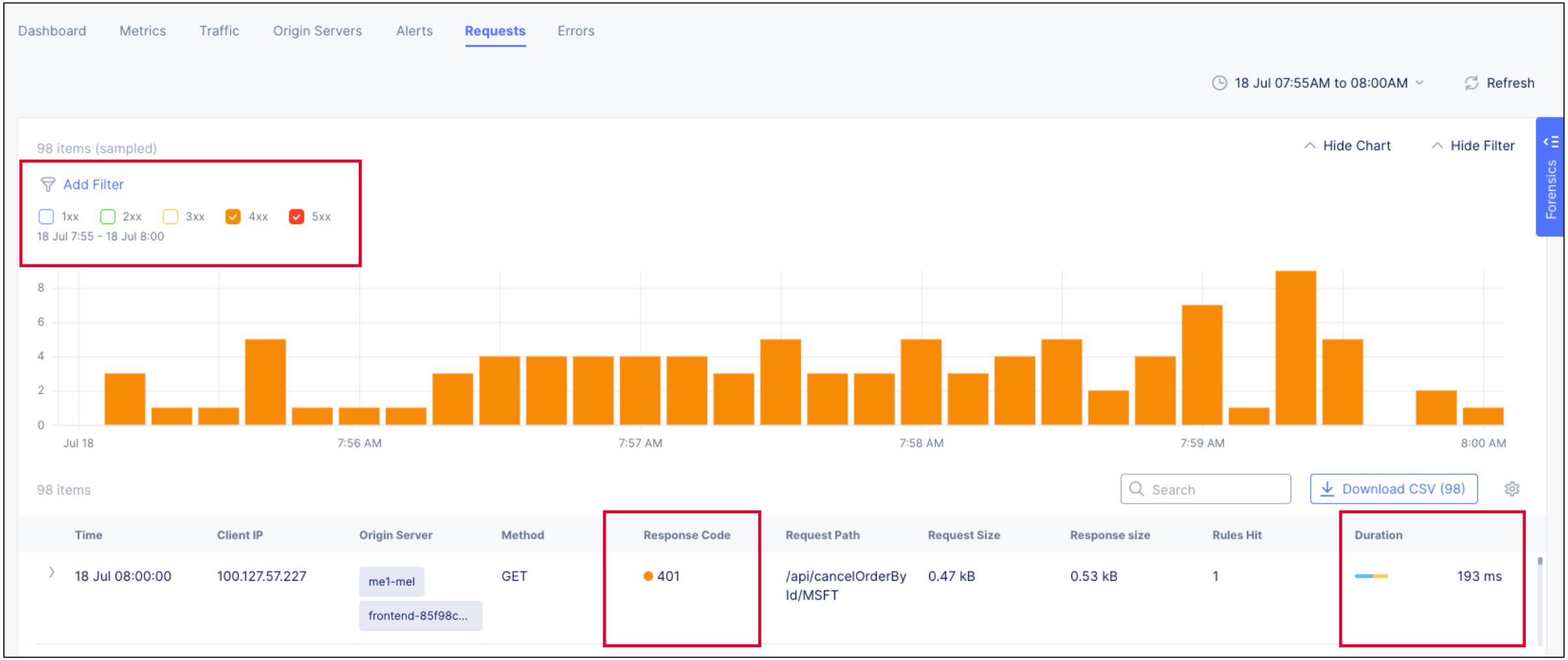
API8: Security Misconfiguration

Protection#04 – Monitoring API Metrics

- *XC WAAP is not just a security product, but it also provides detailed network and app performance data.*
- *With monitoring for error rate or drop rate for the API endpoints, a user can find any potential service issues earlier.*

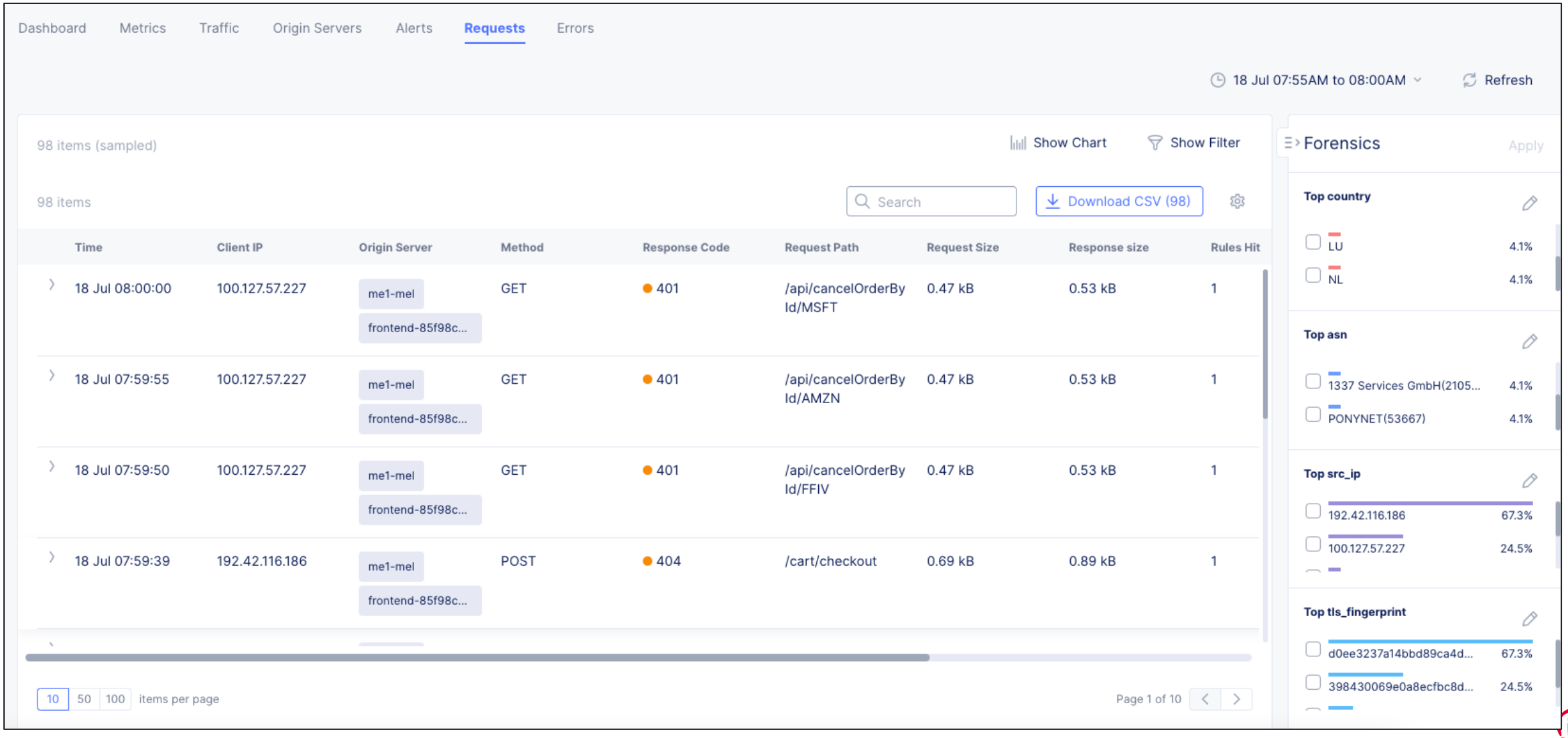
API8: Security Misconfiguration

Protection#04 – Monitoring API Metrics



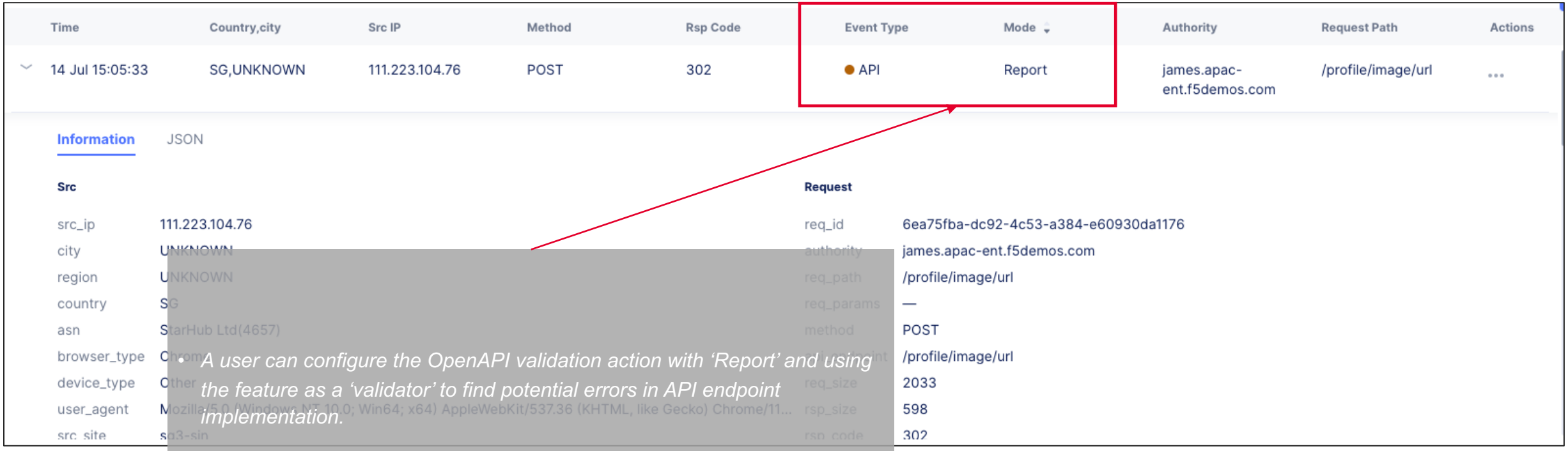
API8: Security Misconfiguration

Protection#04 – Monitoring API Metrics



API8: Security Misconfiguration

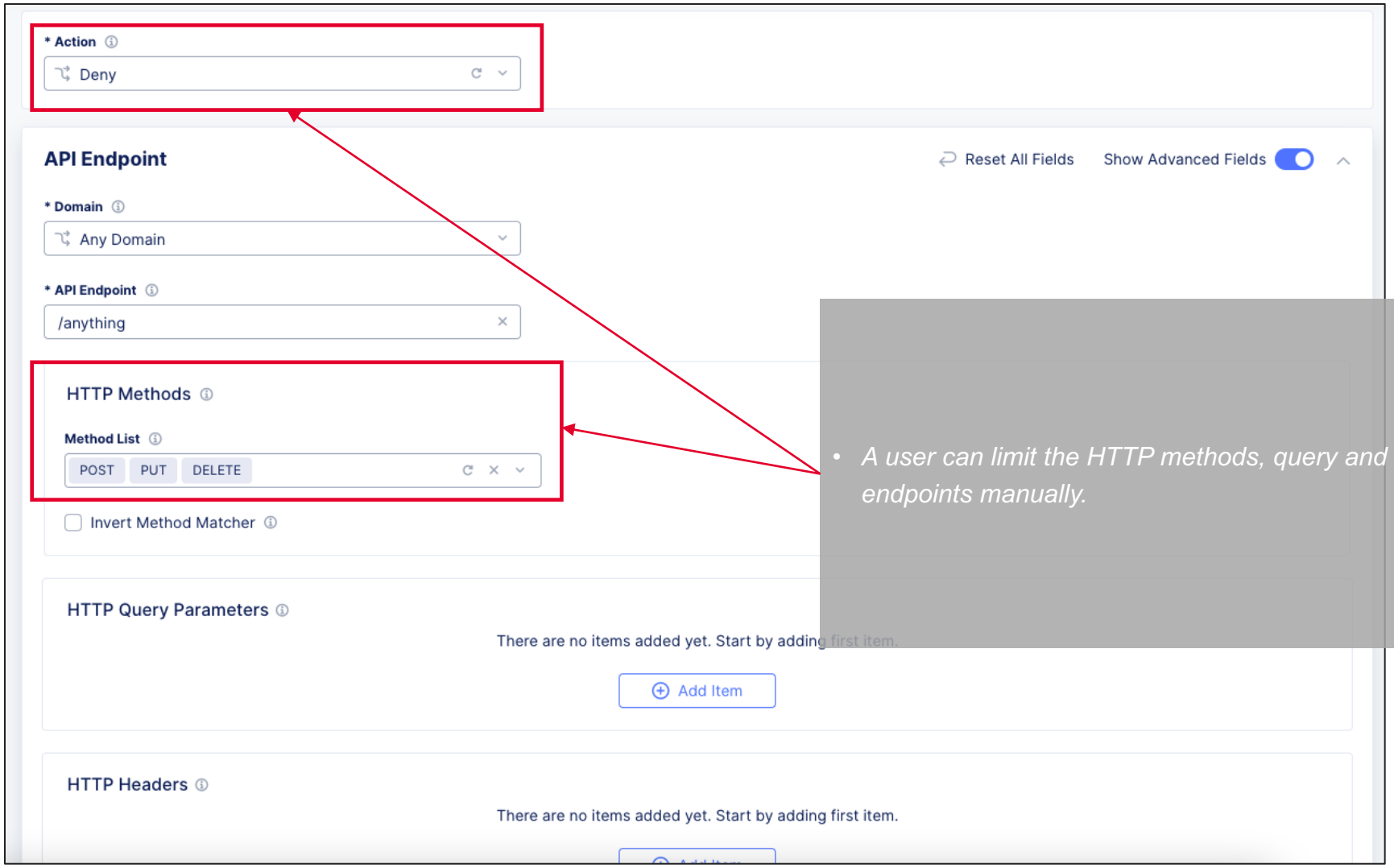
Protection#05 – OpenAPI Validation



- *A user can configure the OpenAPI validation action with 'Report' and using the feature as a 'validator' to find potential errors in API endpoint implementation.*

API8: Security Misconfiguration

Protection#06 – HTTP Verbs Control



- *A user can limit the HTTP methods, query and headers for specific API endpoints manually.*

API8: Security Misconfiguration

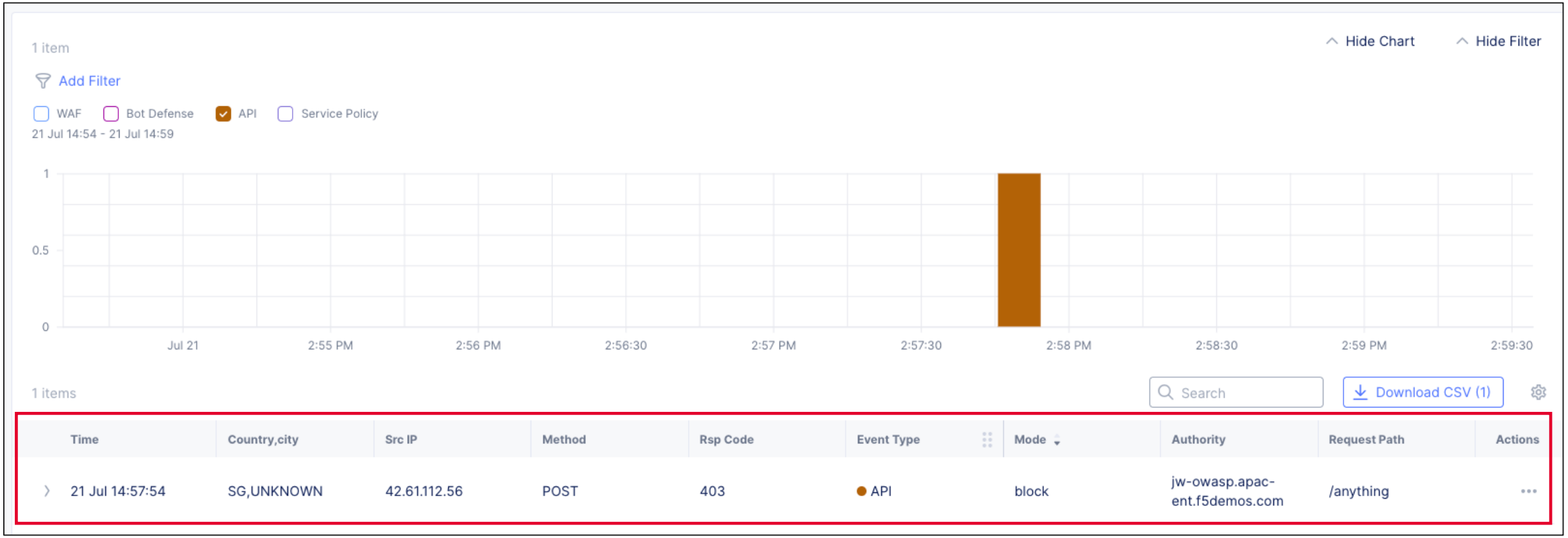
Protection#06 – HTTP Verbs Control

```
j.lee@C02FPFZPMD6M Desktop %
j.lee@C02FPFZPMD6M Desktop % curl -X POST https://jw-owasp.apac-ent.f5demos.com/anything
<html><head><title>Error Page</title></head>
<body>The requested URL was rejected. Please consult with your administrator.<br/><br/>
Your support ID is 2893bba9-7513-4d71-bf27-0febd6ce174b<h2>Error 403 - Forbidden</h2>F5 site: sg3-sin<br/><br/><a href='javascript:history
.back();'>[Go Back]</a></body></html>
j.lee@C02FPFZPMD6M Desktop %
j.lee@C02FPFZPMD6M Desktop %
j.lee@C02FPFZPMD6M Desktop % curl -X GET https://jw-owasp.apac-ent.f5demos.com/anything
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Host": "jw-owasp.apac-ent.f5demos.com",
    "User-Agent": "curl/7.87.0",
    "X-Envoy-External-Address": "42.61.112.56",
    "X-F5-Request-Id": "21e1e5d9-5381-44b5-90c3-ecac0c778c53"
  },
  "json": null,
  "method": "GET",
  "origin": "42.61.112.56",
  "url": "https://jw-owasp.apac-ent.f5demos.com/anything"
}
j.lee@C02FPFZPMD6M Desktop %
```

- *GET is allowed, but POST is blocked.*

API8: Security Misconfiguration

Protection#06 – HTTP Verbs Control



Session 1: The new OWASP Top 10 API Security 2023

API10:2023 – Unsafe Consumption of APIs

API10: Unsafe Consumption of APIs

Developers tend to trust data received from 3rd party APIs, especially for APIs from well-known companies. Thus, it sometimes has a lack of input validation and sanitization for those APIs.

- Does not properly validate and sanitize the data from 3rd party companies.
- Blindly follows redirections.
- No timeout mechanism with 3rd party APIs.
- No limit the number of resources.

Scenario #2

An API integrates with a third-party service provider to safely store sensitive user medical information. Data is sent over a secure connection using an HTTP request like the one below:

```
POST /user/store_phr_record
{
  "genome": "ACTAGTAG__TTGADDAAIICCTT…"
}
```

Bad actors found a way to compromise the third-party API and it starts responding with a `308 Permanent Redirect` to requests like the previous one.

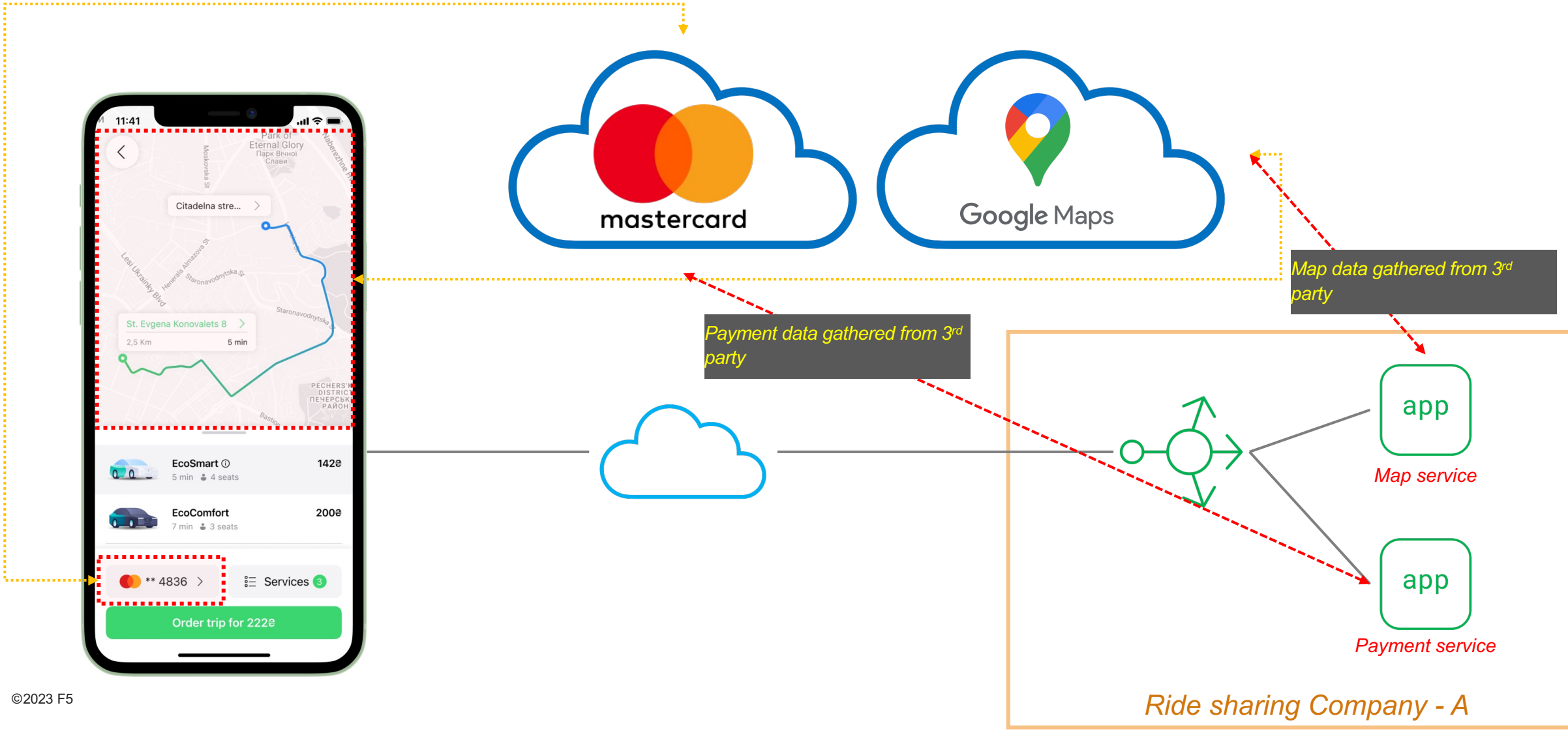
```
HTTP/1.1 308 Permanent Redirect
Location: https://attacker.com/
```

Since the API blindly follows the third-party redirects, it will repeat the exact same request including the user's sensitive data, but this time to the attacker's server.

OWASP API SECURITY TOP 10

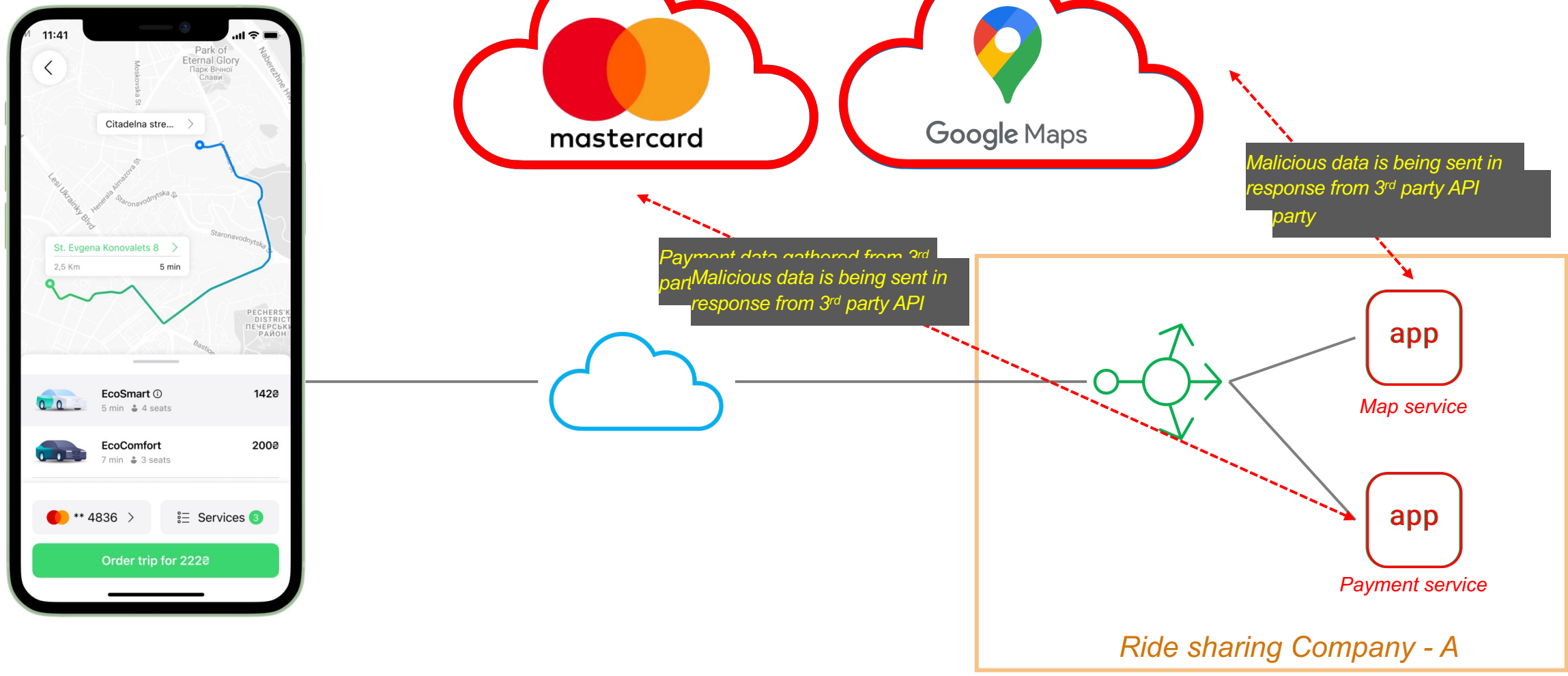
API10: Unsafe Consumption of APIs

How does it work? And why does it risky?

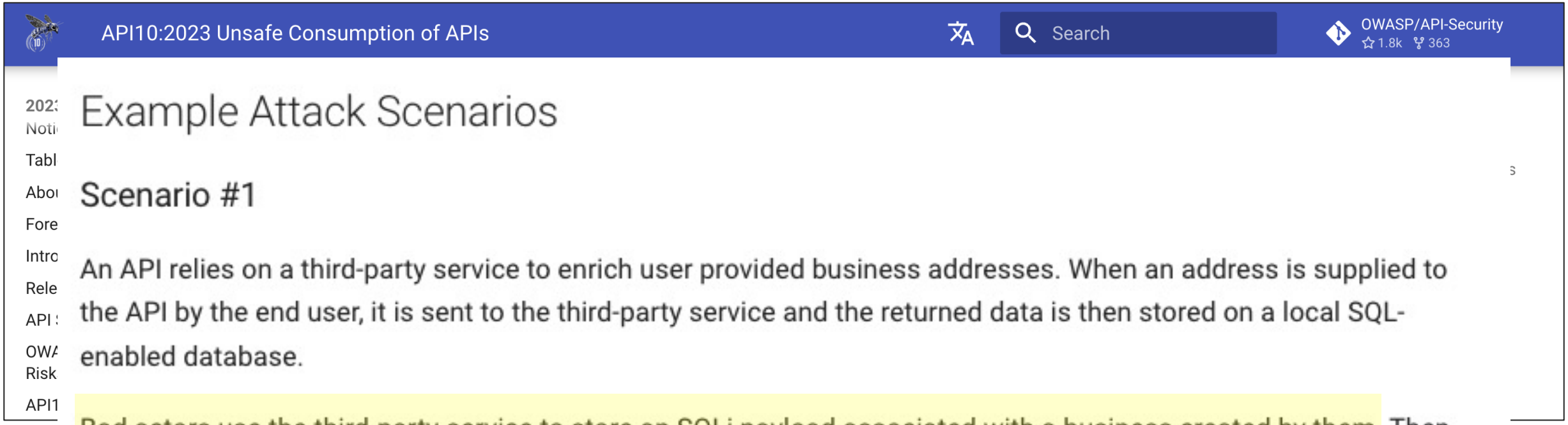


API10: Unsafe Consumption of APIs

How does it work? And why does it risky?

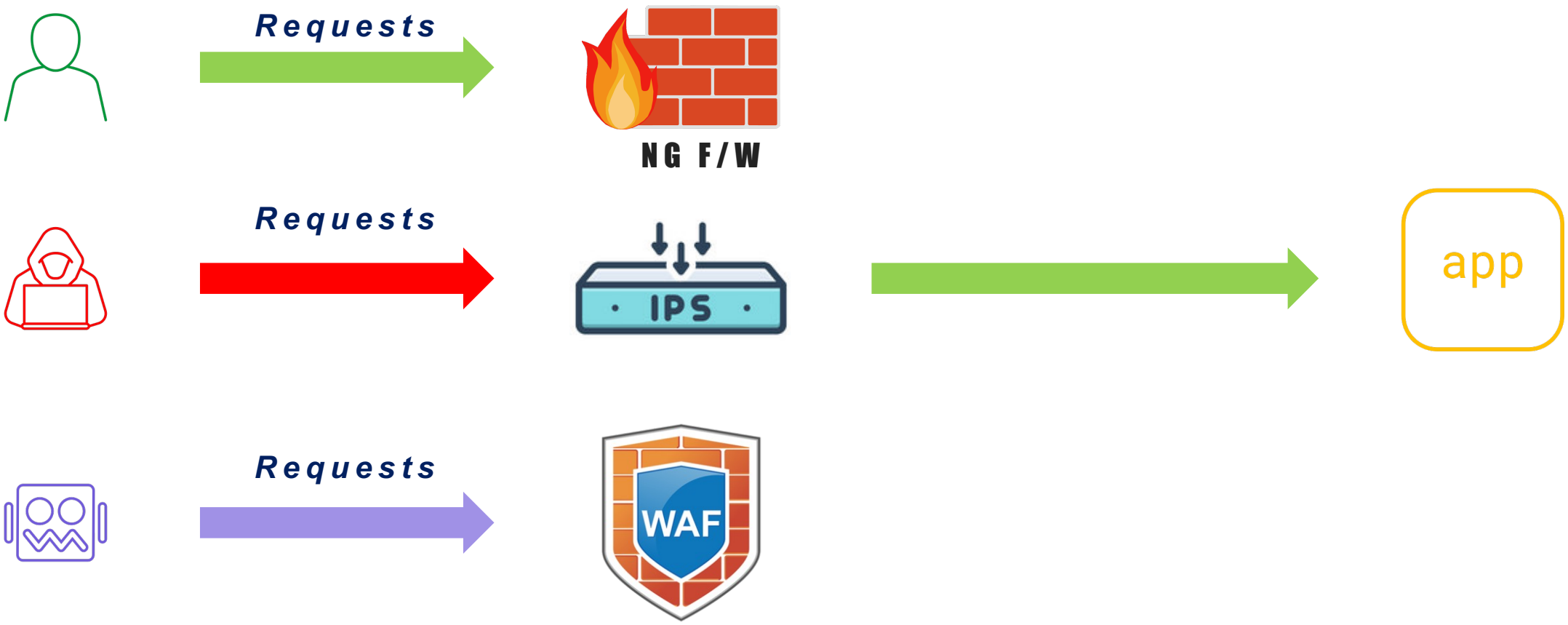


OWASP Recommendation



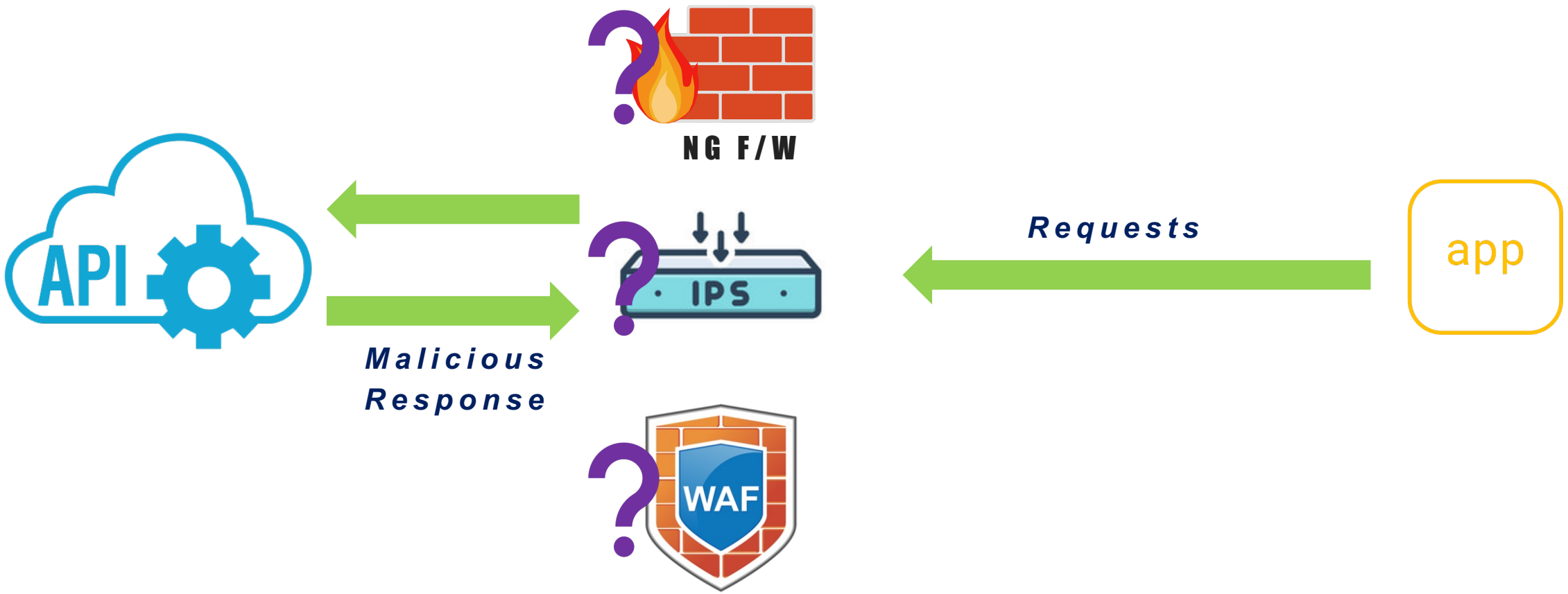
API10: Unsafe Consumption of APIs

How to Stop the ‘Unsafe Consumption of APIs’



API10: Unsafe Consumption of APIs

How to Stop the ‘Unsafe Consumption of APIs’



API10: Unsafe Consumption of APIs

How to Stop the ‘Unsafe Consumption of APIs’

Requests

REQUEST-932-APPLICATION-ATTACK-RCE.conf REQUEST-933-APPLICATION-ATTACK-PHP.conf REQUEST-934-APPLICATION-ATTACK-GENERIC.conf

TODO

REQUEST-941-APPLICATION-ATTACK-XSS.conf TODO

| **REQUEST-942-APPLICATION-ATTACK-SQLI** | Configuration Path: `rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf` | | Within this configuration file we provide rules that protect against SQL injection attacks. SQLi attackers occur when an attacker passes crafted control characters to parameters to an area of the application that is expecting only data. The application will then pass the control characters to the database. This will end up changing the meaning of the expected SQL query.

| **REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION** | Configuration Path: `rules/REQUEST-43-APPLICATION-ATTACK-SESSION-FIXATION.conf` | | These rules focus around providing protection against Session Fixation attacks.

Requests

| **RESPONSE-954-DATA-LEAKAGES-IIS** | Configuration Path: `rules/RESPONSE-954-DATA-LEAKAGES-IIS.conf` | | These rules provide protection against data leakages that may occur because of Microsoft IIS

| **RESPONSE-952-DATA-LEAKAGES-JAVA** | Configuration Path: `rules/RESPONSE-952-DATA-LEAKAGES-JAVA.conf` | | These rules provide protection against data leakages that may occur because of Java

| **RESPONSE-953-DATA-LEAKAGES-PHP** | Configuration Path: `rules/RESPONSE-953-DATA-LEAKAGES-PHP.conf` | | These rules provide protection against data leakages that may occur because of PHP

| **RESPONSE-950-DATA-LEAKAGES** | Configuration Path: `rules/RESPONSE-950-DATA-LEAKAGES.conf` | | These rules provide protection against data leakages that may occur generiсly

API10: Unsafe Consumption of APIs

How to Stop the ‘Unsafe Consumption of APIs’

Secure Proxy for External API calls

- With proxy configuration with the custom script, reverse proxy can validate and sanitize the response data from external APIs.

OpenAPI Spec Validation

- With OpenAPI validation process, you can validate the response from 3rd party API vendors.

API10: Unsafe Consumption of APIs

Protection#01 – Secure Proxy for External API calls

```
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ curl -k -H 'Content-Type: application/json' https://14c1fa7d-c460-4da3-ba49-a923dbb3a63d.access.udf.f5.com/normal.json
{
    "email": "james@test.com",
    "age": "32",
    "company": "F5"
}
ubuntu@ubuntu:~$
```

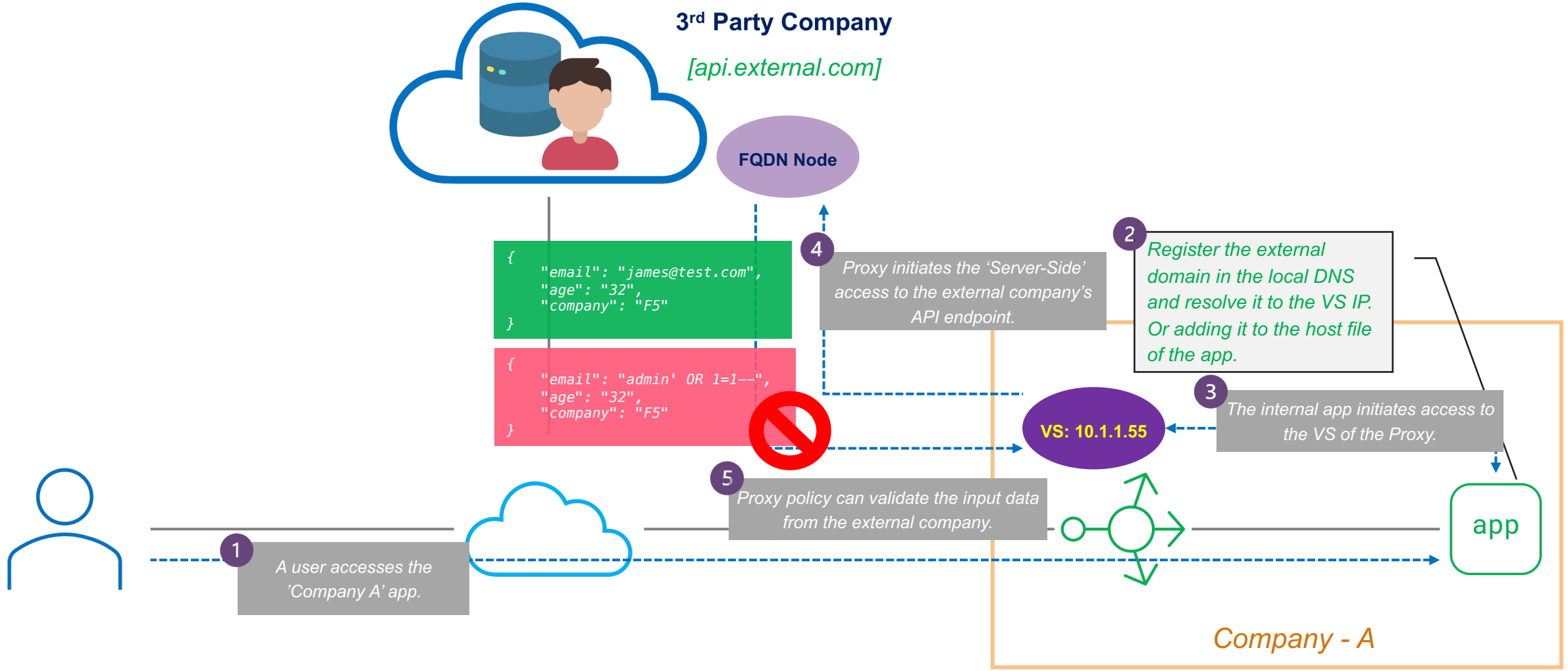
- *The internal app can fetch the user's data from the external company via API.*

```
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ curl -k -H 'Content-Type: application/json' https://14c1fa7d-c460-4da3-ba49-a923dbb3a63d.access.udf.f5.com/malformed.json
{
    "email": "admin' OR 1=1--",
    "age": "32",
    "company": "F5"
}
ubuntu@ubuntu:~$
```

- *If the external company's app is compromised, it could send malicious content.*
- *In this example, the attacker put the SQLi payload in the 'email' field. If the internal app trusts the 3rd party company's data without additional validation, this could cause the SQLi attack.*
- *This example is the same example that OWASP described as 'Scenario #1'.*

API10: Unsafe Consumption of APIs

Protection#01 – Secure Proxy for External API calls



API10: Unsafe Consumption of APIs

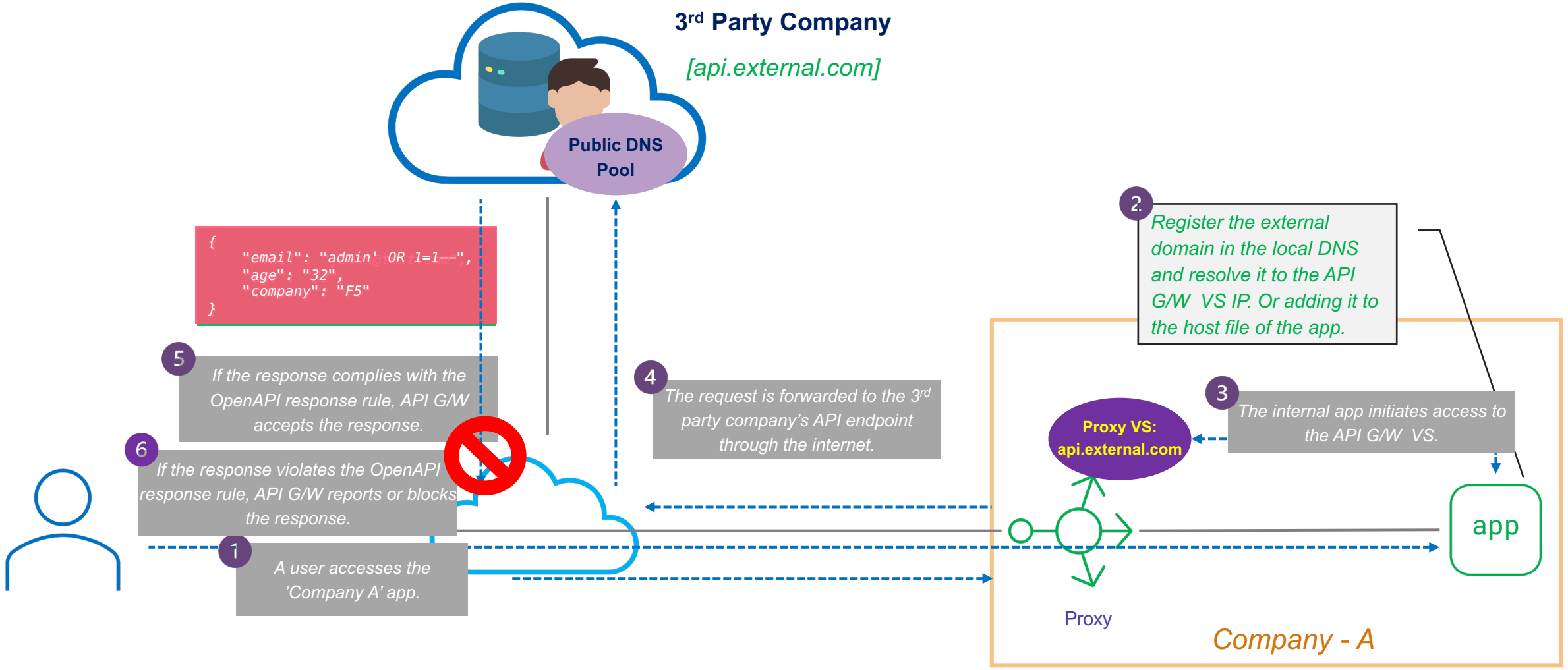
Protection#01 – Secure Proxy for External API calls

```
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ curl -k -H 'Content-Type: application/json' https://14c1fa7d-c460-4da3-ba49-a923dbb3a63d.access.udf.f5.com/normal.json
{
    "email": "james@test.com",
    "age": "32",
    "company": "F5"
}
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ curl -k -H 'Content-Type: application/json' https://14c1fa7d-c460-4da3-ba49-a923dbb3a63d.access.udf.f5.com/malformed.json
curl: (56) OpenSSL SSL_read: Connection reset by peer, errno 104
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$
```

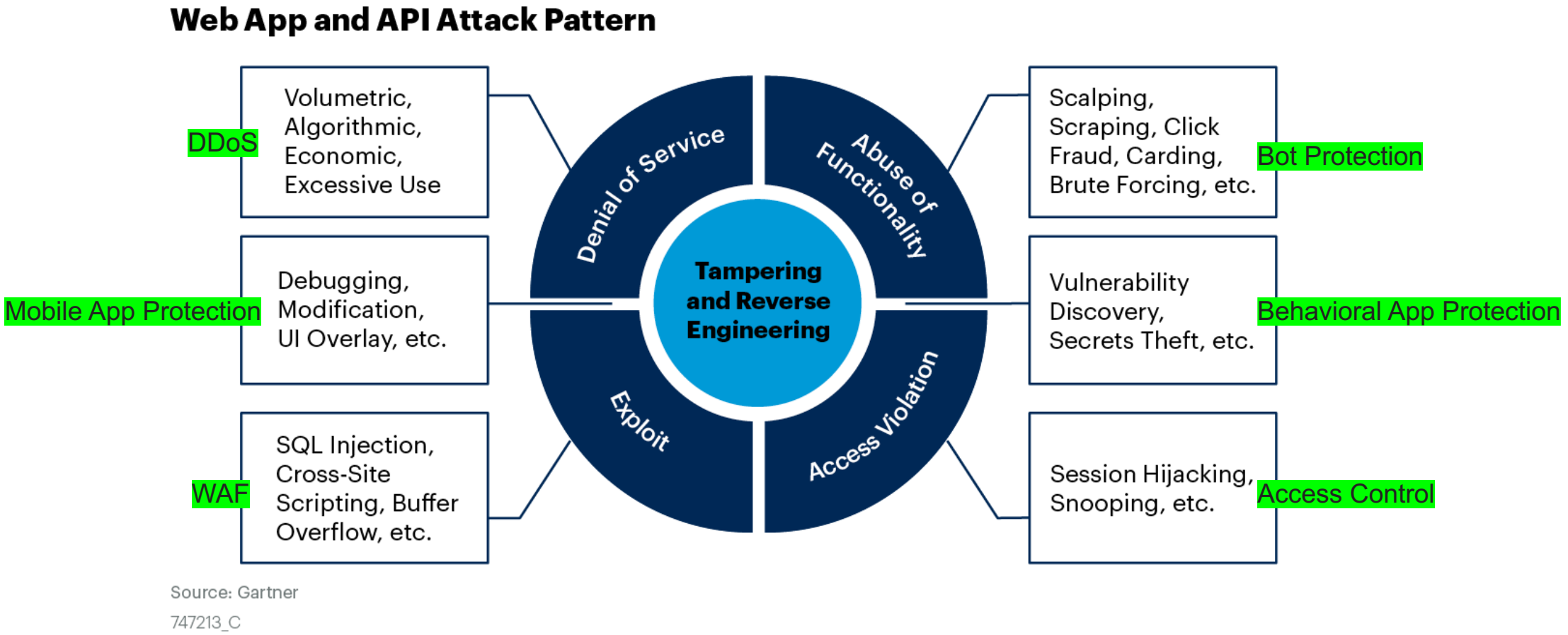
- *If the received data is aligned with the defined format in the Proxy policy, the data is consumed as normal.*
- *However, if the data format violates the defined format in the Proxy policy, Proxy reset the session.*

API10: Unsafe Consumption of APIs

Protection#02 – OpenAPI Validation



Web App and API Attack Pattern



Gartner