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.
NEW OWASP API TOP 10 - 2023

OWASP API Top 10 - 2019

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resource & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

Access Control
Network Security
App Security
NEW OWASP API TOP 10 - 2023

OWASP API Top 10 - 2023

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resource & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

- Broken Object Level Authorization
- Broken Authentication
- Broken Object Property Level Authorization
- Unrestricted Resource Consumption
- Broken Function Level Authorization
- Unrestricted Access to Sensitive Business Flows
- Server Side Request Forgery
- Security Misconfiguration
- Improper Inventory Management
- Unsafe Consumption of APIs
NEW OWASP API TOP 10 - 2023

OWASP API Top 10 - 2023

API1. Broken Object Level Authorization
API2. Broken Authentication
API3. Broken Object Property Level Authorization
API4. Unrestricted Resource Consumption
API5. Broken Function Level Authorization
API7. Server Side Request Forgery
API8. Security Misconfiguration
API9. Improper Inventory Management
API10. Unsafe Consumption of APIs

Access Control
Network Security
App Security
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

Follow Principles

- Do not underestimate weak configs
- Always verify

Use proactive security when it's available

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

Do not

- Verify the data and privilege. Even the one from the trusted partners
- API key, Human/Non-human detection and OpenAPI validation

Block ToR IPs, CORS configuration, redirection handlings and etc.
API Security Maturity Model
API Security Maturity Model

Gartner’s API Strategy Maturity Model

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.

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 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.
Understanding the importance of the ‘API Security’

API G/W Traffic management Basic Authentication(API Key)

Predictive Security

Proactive Security

OAuth Access Control

Reactive Security

- API Discovery
- Bot Protection
- WAF Protection

Behavioral detection - Using A.I./ML detection

API SECURITY MATURITY MODEL

API Security Maturity Model

Level 1 Level 2 Level 3 Level 4 Level 5

Predictive Security

Proactive Security

Reactive Security

Level 1

Level 2

Level 3

Level 4

Level 5

Understanding the importance of the ‘API Security’
## API Security Maturity Model

### Reactive vs Proactive vs Predictive

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

API Security Maturity Model

- Reactive Security
- Proactive Security
- Predictive Security

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
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

<table>
<thead>
<tr>
<th>Case#01.</th>
<th>Case#02.</th>
<th>Case#03.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- An attacker finds the <code>.bash_history</code> file under the root directory of the server, which contains commands used by the DevOps team to access the API.</td>
<td>- An attacker finds some API endpoints that are accessible from the internet but don’t have the authentication mechanism.</td>
<td>- Some API endpoints include potentially critical data in their token or header or body, and an attacker can exploit those data potentially.</td>
</tr>
</tbody>
</table>
API8: Security Misconfiguration
How to Stop the Security Misconfiguration

- 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.

Continuous Monitoring
• Continuous assessment of the security effectiveness of API endpoints.

TLS Policy Setting
• Need to have the capability to set up the different TLS policies based on the backend apps’ requirements.

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

HTTP Methods Control
• Enable only relevant HTTP methods for the specific API endpoints.

CORS Policy
• Need to configure the correct CORS settings in your apps.
API8: Security Misconfiguration

Protection#01 – CORS Configuration

The webserver has a vulnerable CORS setting with the wildcard ("*").
### CORS Policy Configuration

<table>
<thead>
<tr>
<th>Order</th>
<th>Allow Origin RegEx</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>https://([a-z0-9]+).f5.com</td>
</tr>
<tr>
<td>2</td>
<td>https://([a-z0-9]+).nginx.com</td>
</tr>
</tbody>
</table>

- 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

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;
set-cookie: 8bd883=113pxIAYZfs3NC8i17X1ZxbirrP63rHR02105yCXnyH9AbGS5zCspKuBxk/mVdoxe53jijQsnwzmTG+gqBy34XgbZ6YDUaIo0FHRL1; 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.
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@C02FFPZPMD6M ~ %
j.lee@C02FFPZPMD6M ~ % 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 hello (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
  * issuer: C=US; O=Let's Encrypt; CN=R3
* SSL certificate verify ok.
```

• TLS v1.0 works correctly.

"jw-owasp.apac-ent.f5demos.com"
• Setting the TLS level to ‘High’ means TLSv1.2+ and PFS will be only allowed.
API8: Security Misconfiguration

Protection#02 – TLS Configuration

```
j.lee@C02FPFZPM06M ~ %
j.lee@C02FPFZPM06M ~ % 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
  * CApas: 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@C02FPFZPM06M ~ %
j.lee@C02FPFZPM06M ~ %
```

- TLS v1.0 is NOT allowed in XC WAAP.
API8: Security Misconfiguration

Protection#02 – TLS Configuration

• A user also can custom TLS settings in XC WAAP.
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

- 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’.

<table>
<thead>
<tr>
<th>API Endpoint</th>
<th>Method</th>
<th>Sensitive Data</th>
<th>Threat Level</th>
<th>Authentication S...</th>
<th>Authentication T...</th>
<th>API Category</th>
<th>Risk Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/auth/oidc</td>
<td>GET</td>
<td>—</td>
<td>None</td>
<td>Authenticated</td>
<td>JWT-Header</td>
<td>Discovered</td>
<td>Shadow</td>
</tr>
<tr>
<td>/api/getPaymentInfo/me</td>
<td>GET</td>
<td>Credentials</td>
<td>None</td>
<td>Un-Authenticated</td>
<td>—</td>
<td>Discovered</td>
<td>Shadow</td>
</tr>
<tr>
<td>/trading/login.php</td>
<td>GET</td>
<td>—</td>
<td>High</td>
<td>Un-Authenticated</td>
<td>—</td>
<td>Discovered</td>
<td>Shadow</td>
</tr>
<tr>
<td>/api/getStock</td>
<td>GET</td>
<td>—</td>
<td>None</td>
<td>Un-Authenticated</td>
<td>—</td>
<td>Discovered</td>
<td>Shadow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Un-Authenticated</td>
<td>—</td>
<td>Discovered</td>
<td>Shadow</td>
</tr>
</tbody>
</table>
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 Interface Screenshot](image)

## Table: API Requests

<table>
<thead>
<tr>
<th>Time</th>
<th>Client IP</th>
<th>Origin Server</th>
<th>Method</th>
<th>Response Code</th>
<th>Request Path</th>
<th>Request Size</th>
<th>Response Size</th>
<th>Rules Hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Jul 08:00</td>
<td>100.127.57.227</td>
<td>met1-mei</td>
<td>GET</td>
<td>401</td>
<td>/api/cancelOrderById/MSFT</td>
<td>0.47 kB</td>
<td>0.53 kB</td>
<td>1</td>
</tr>
<tr>
<td>18 Jul 07:59:55</td>
<td>100.127.57.227</td>
<td>met1-mei</td>
<td>GET</td>
<td>401</td>
<td>/api/cancelOrderById/AMZN</td>
<td>0.47 kB</td>
<td>0.53 kB</td>
<td>1</td>
</tr>
<tr>
<td>18 Jul 07:59:50</td>
<td>100.127.57.227</td>
<td>met1-mei</td>
<td>GET</td>
<td>401</td>
<td>/api/cancelOrderById/FFIV</td>
<td>0.47 kB</td>
<td>0.53 kB</td>
<td>1</td>
</tr>
<tr>
<td>18 Jul 07:59:30</td>
<td>102.42.116.186</td>
<td>met1-mei</td>
<td>POST</td>
<td>404</td>
<td>/cart/checkout</td>
<td>0.09 kB</td>
<td>0.89 kB</td>
<td>1</td>
</tr>
</tbody>
</table>

© 2023 F5
A user can configure the OpenAPI validation action with ‘Report’ and use the feature as a ‘validator’ to find potential errors in API endpoint implementation.
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@C02FPFZPM6M 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/>
Your support ID is 2893bb9a-7513-4d71-bf27-0febd6ce174b<br/>
Error 403 - Forbidden<br/>
F5 site: sg3-sin<br/>
<a href='javascript:history.back();'>[Go Back]</a></body></html>
j.lee@C02FPFZPM6M Desktop %
j.lee@C02FPFZPM6M 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@C02FPFZPM6M Desktop %
```

• GET is allowed, but POST is blocked.
## API8: Security Misconfiguration

### Protection#06 – HTTP Verbs Control

<table>
<thead>
<tr>
<th>Time</th>
<th>Country, city</th>
<th>Src IP</th>
<th>Method</th>
<th>Resp Code</th>
<th>Event Type</th>
<th>Mode</th>
<th>Authority</th>
<th>Request Path</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>21 Jul 14:57:54</td>
<td>SG,UNKNOWN</td>
<td>42.61.112.56</td>
<td>POST</td>
<td>403</td>
<td>API</td>
<td>block</td>
<td>jw-owasp.apac-ent.f5demos.com</td>
<td>anything</td>
<td>???</td>
</tr>
</tbody>
</table>
Session 1: The new OWASP Top 10 API Security 2023

API10:2023 – 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...TTGA00AA11CCCTT..."
}
```

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.
API10: Unsafe Consumption of APIs

How does it work? And why does it risky?

Ride sharing Company - A

Map service

Payment service

Map data gathered from 3rd party

Payment data gathered from 3rd party
API10: Unsafe Consumption of APIs

How does it work? And why does it risky?

**Compromised**

Malicious data is being sent in response from 3rd party API

Payment data gathered from 3rd party API

Map service

Payment service

Ride sharing Company - A
Example Attack Scenarios

Scenario #1

An API relies on a third-party service to enrich user provided business addresses. When an address is supplied to the API by the end user, it is sent to the third-party service and the returned data is then stored on a local SQL-enabled database.

Bad actors use the third-party service to store an SQLi payload associated with a business created by them. Then they go after the vulnerable API providing specific input that makes it pull their "malicious business" from the third-party service. The SQLi payload ends up being executed by the database, exfiltrating data to an attacker's controlled server.
API10: Unsafe Consumption of APIs

How to Stop the 'Unsafe Consumption of APIs'

1. Requests
2. NG F/W
3. IPS
4. WAF
5. app
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'

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-943-APPLICATION-ATTACK-SESSION-FIXATION.conf |
| These rules focus around providing protection against Session Fixation attacks.

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 generically
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

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

- 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

1. A user accesses the 'Company A' app.

3rd Party Company

[api.external.com]

FQDN Node

{ "email": "james@test.com", "age": "32", "company": "F5" }

Proxy initiates the 'Server-Side' access to the external company's API endpoint.

Proxy policy can validate the input data from the external company.

Register the external domain in the local DNS and resolve it to the VS IP. Or adding it to the host file of the app.

The internal app initiates access to the VS of the Proxy.

Company - A

app

VS: 10.1.1.55
API10: Unsafe Consumption of APIs

Protection#01 – Secure Proxy for External API calls

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.

```bash
ubuntu@ubuntu:~$ curl -k -H 'Content-Type: application/json' https://14c1fa7d-c460-4da3-ba49-a923dab3a63d.access.udf.f5.com/malformed.json
```
curl: (56) OpenSSL SSL_read: Connection reset by peer, errno 104

```bash
ubuntu@ubuntu:~$
```
Register the external domain in the local DNS and resolve it to the API G/W VS IP. Or adding it to the host file of the app.

Public DNS Pool

3rd Party Company

[api.external.com]

A user accesses the 'Company A' app.

The request is forwarded to the 3rd party company’s API endpoint through the internet.

If the response complies with the OpenAPI response rule, API G/W accepts the response.

If the response violates the OpenAPI response rule, API G/W reports or blocks the response.

{ "email": "admin' OR 1=1--", "age": "32", "company": "F5"

If the response complies with the OpenAPI response rule, API G/W accepts the response.

{ "email": "james@test.com", "age": "32", "company": "F5"

Proxy VS: api.external.com

Company - A
Web App and API Attack Pattern

- **DDoS**
  - Volumetric, Algorithmic, Economic, Excessive Use

- **Mobile App Protection**
  - Debugging, Modification, UI Overlay, etc.

- **WAF**
  - SQL Injection, Cross-Site Scripting, Buffer Overflow, etc.

- **Bot Protection**
  - Scalping, Scraping, Click Fraud, Carding, Brute Forcing, etc.

- **Behavioral App Protection**
  - Vulnerability Discovery, Secrets Theft, etc.

- **Access Control**
  - Session Hijacking, Snooping, etc.

Source: Gartner
747213_C