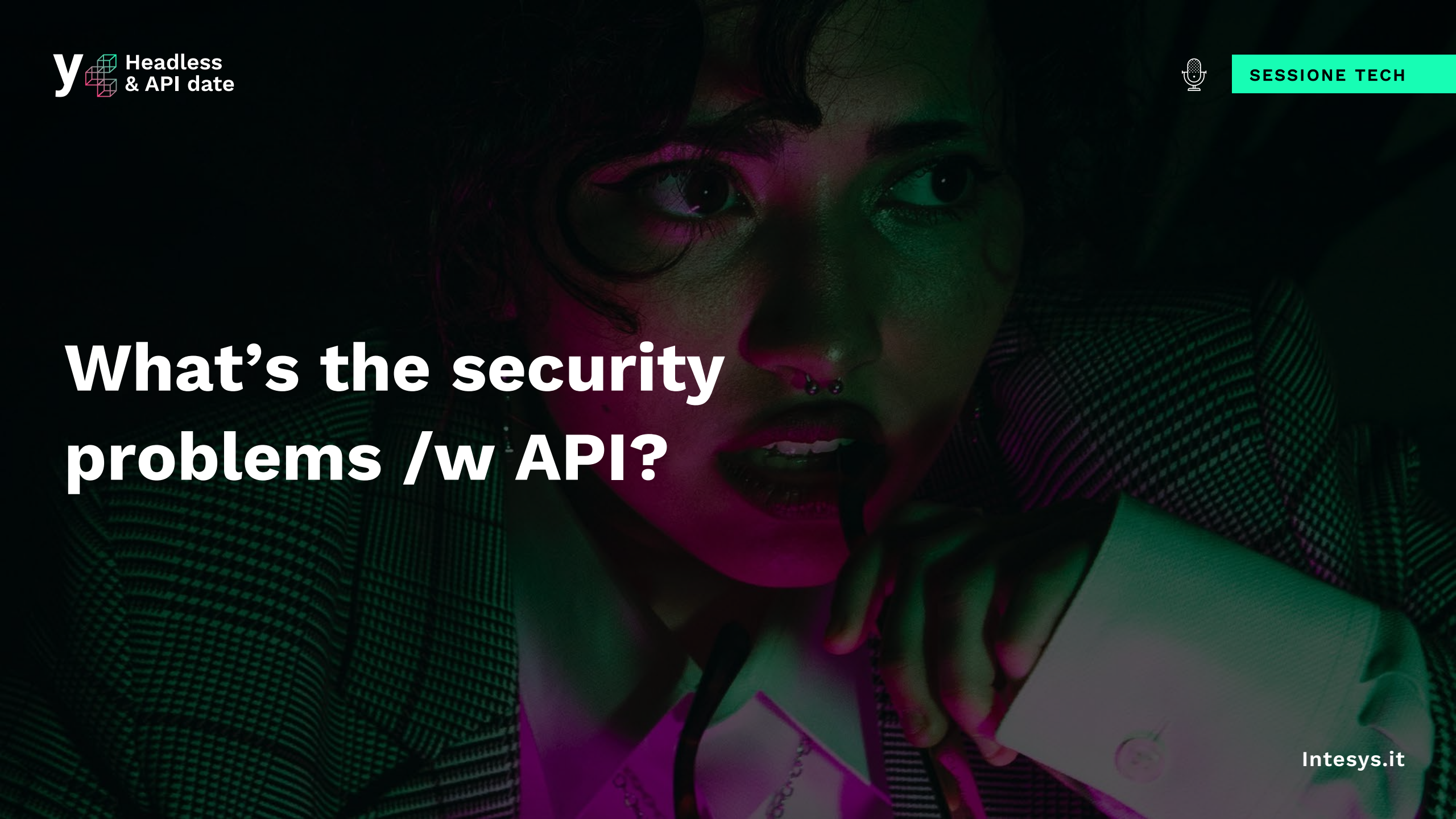
Headless & API date | Liferay | mia Platform | Intesys | SESSIONE TECH

API Security: come proteggersi dalle 10 minacce più frequenti (OWASP)

DENIS SIGNORETTO
IT ARCHITECT, INTESYS

Headless & API date

SESSIONE TECH



What’s the security problems /w API?

Intesys.it


Headless
& API date
SESSIONE TECH
How can I mitigate
the risks?
Intesys.it

OWASP API Security Top 10

1. Broken Object Level Authorization
2. Broken User Authentication
3. Excessive Data Exposure
4. Lack of Resources & Rate Limiting
5. Broken Function Level
6. Mass Assignment
7. Security Misconfiguration
8. Injection
9. Improper Assets Management
10. Insufficient Logging & Monitoring

OWASP API Security Top 10

- Authentication/Authorization
- Data Protection
- Governance/Operations

*Categorization according to Isabelle Mauny of 42 Crunch

1. Broken Object Level Authorization
2. Broken User Authentication
3. Excessive Data Exposure
4. Lack of Resources & Rate Limiting
5. Broken Function Level
6. Mass Assignment
7. Security Misconfiguration
8. Injection
9. Improper Assets Management
10. Insufficient Logging & Monitoring

Headless & API date

SESSIONE TECH

Authentication & Authorization

Intesys.it

Headless & API date

SESSIONE TECH

API 02
Broken User Authentication

Intesys.it

Use case

Authentication mechanisms are implemented incorrectly.

Allow attackers to compromise authentication tokens or to exploit implementation flaws to **assume other user's identities** temporarily or permanently.

Compromising system's ability to identify the client/user, **compromises API security overall**.

02 – Broken Authentication

02 – Broken Authentication

How to prevent

- Use standard authentication protocols like **OAuth2** & **OpenIdConnect**
- Use short-lived access tokens
- **Multi-factor authentication**
- Authenticate your apps (so you know who is talking to you)
- Use **stricter rate-limiting** for **authentication**, implement **lockout policies** and **weak password checks**

- Check all possible ways to authenticate to all APIs

(!) Test authentication with all kind of combinations

Headless & API date

SESSIONE TECH

API 01

Broken Object Level Authorization

Intesys.it

Use case

Attacker **substitutes ID of their resource** in API call with **an ID of a resource belonging to another user**.

Lack of proper **authorization checks** allows access.

Represent in about 40% of all API attacks

https://salt.security/blog/owasp-api-security-top-10-explained

01 – BOLA Broken Object Level Authorization

01 – BOLA Broken Object Level Authorization

How to prevent

- Don’t rely on IDs sent from client
- **Implement authorization checks** with **user policies** and in **every controller layer (onion approach)**

- Avoid physical IDs or serial IDs (675, 676, 678,...). Use random or non-guessable IDs (like UUIDs)

(!) Test this use case

Headless & API date

SESSIONE TECH

API 05
Broken Function Level

Intesys.it

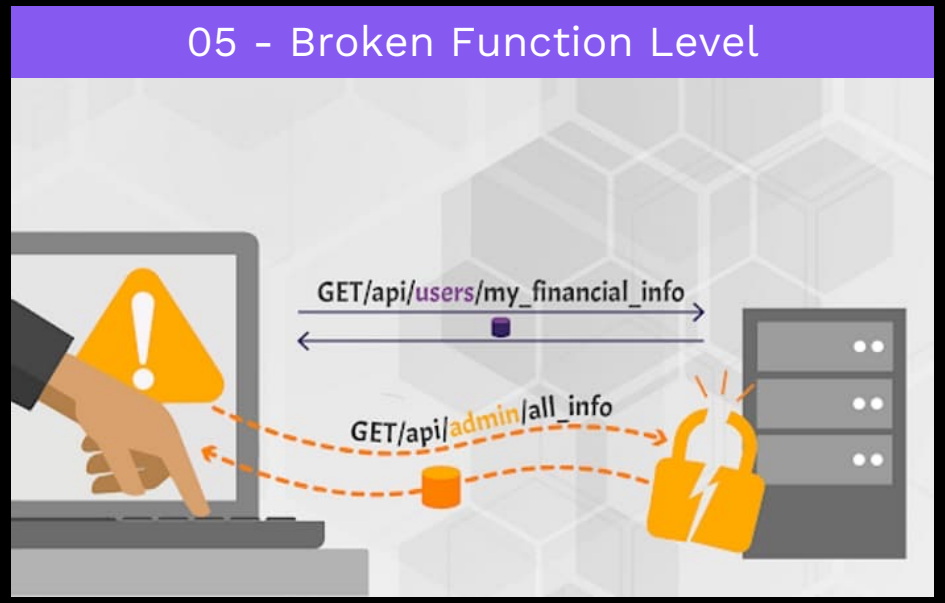
Use case

API relies on client to use user level or admin level APIs.

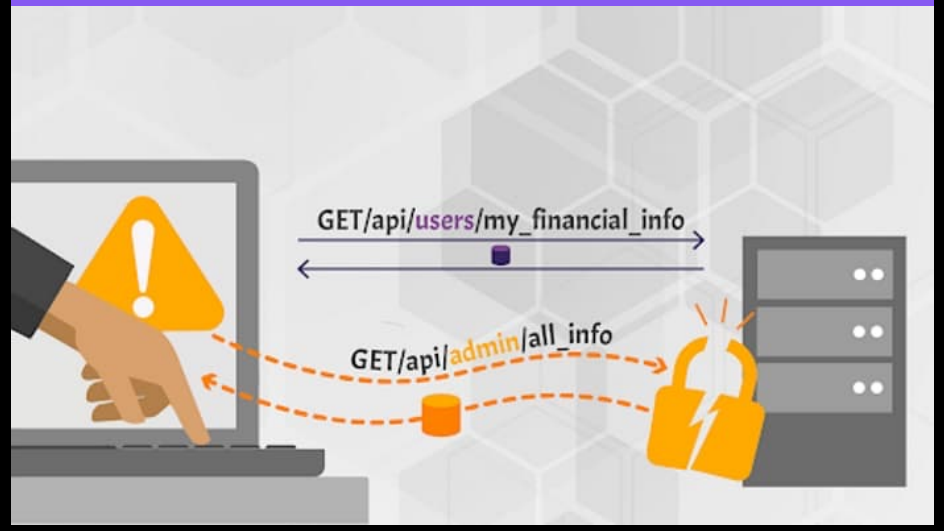
Attacker figures out the “hidden” admin/unauthorized and un protected API methods and invokes them directly

Can be a matter of knowing the URL:

- /api/**users**/v1/user/my_financial_info (authorized)
- /api/**admins**/v1/users/all_info (hidden, not authorized and not protected)



05 - Broken Function Level



How to prevent

- Do not rely on the client, to enforce admin access
- **Design properly your authorization policies**
  - OAuth scopes can help here
- Deny all access by default
- RBAC, only **allow operations to users belonging to the appropriate group or role**
- Whenever possible separate admin and non admin operation, or avoid admin/non-admin on the same API
- Restrict access to admin API
  - By Mutual TLS, IP Range

(!) Test this use case

Headless & API date

SESSIONE TECH

Data Protection

Intesys.it

API 04

Lack of Resources & Rate Limiting

Use case

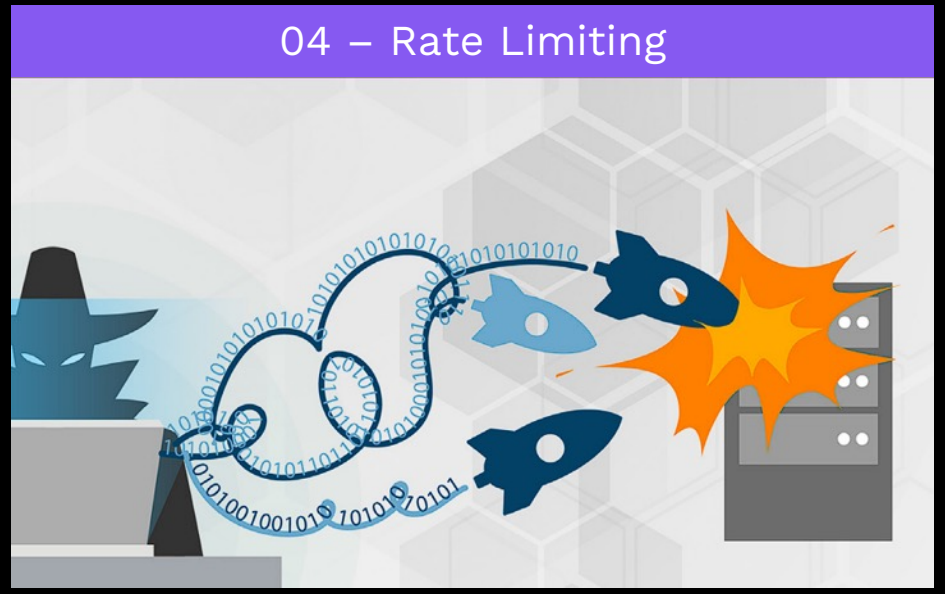
API is not protected against an **excessive amount of calls** or payload sizes.

An API client can make thousands API calls and **the server will still try to fulfill all requests**.

Attackers use that for **DoS** and **brute force attacks**

Attackers can **fire up large number of requests** to harvest data

```
https://api.example.com/v1.1/profile/email/view?user_id=123
https://api.example.com/v1.1/profile/email/view?user_id=124
https://api.example.com/v1.1/profile/email/view?user_id=125
...
...
...
https://api.example.com/v1.1/profile/email/view?user_id=2345
```



04 – Rate Limiting

How to prevent

Mitigate data scrapping by putting rate limiting in place (and alerting!)

- **Implement a limit on how often a client can call the API within a defined timeframe**
- Appropriate rate and resource limit for each functionality

- **Notify the client when the limit is exceeded**
- Limits on “containerized resources” (CPU, RAM, etc.)

04 – Rate Limiting

How to prevent

- **Payload / page size limits as well**

Legitimate – max_return and page_size request attributes are normal	**Attack** – Attackers modify the request to return an abnormally high response size
POST /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0 Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4	POST /example/api/v1/provision/user/search HTTP/1.1 User-Agent: AHC/1.0 Connection: keep-alive Accept: */* Content-Type: application/json; charset=UTF-8 Content-Length: 131 X-Forwarded-For: 10.93.23.4

{
"search_filter":
"user_id=exampleId_100",
"max_return": "250",
"page_size": "250",
"return_attributes": [
 	
]
}
	{
	"search_filter":
	"user_id=exampleId_100",
	"max_return": "20000",
	"page_size": "20000",
	"return_attributes": [
	
	]
	}

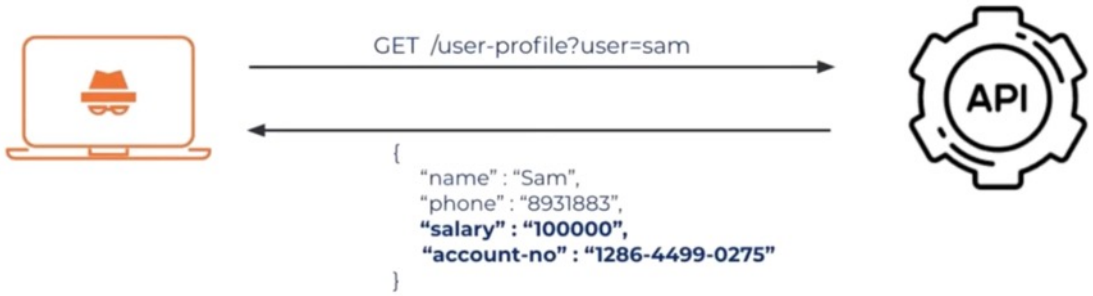
API 03
Excessive Data Exposure

Use case

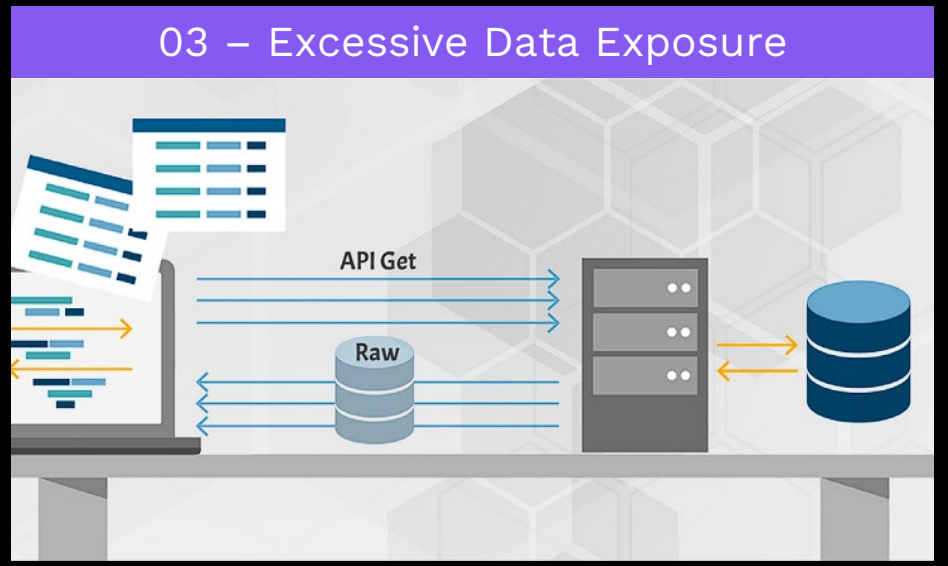
API exposing a lot more data than the client legitimately needs,

relying on the client to do the filtering

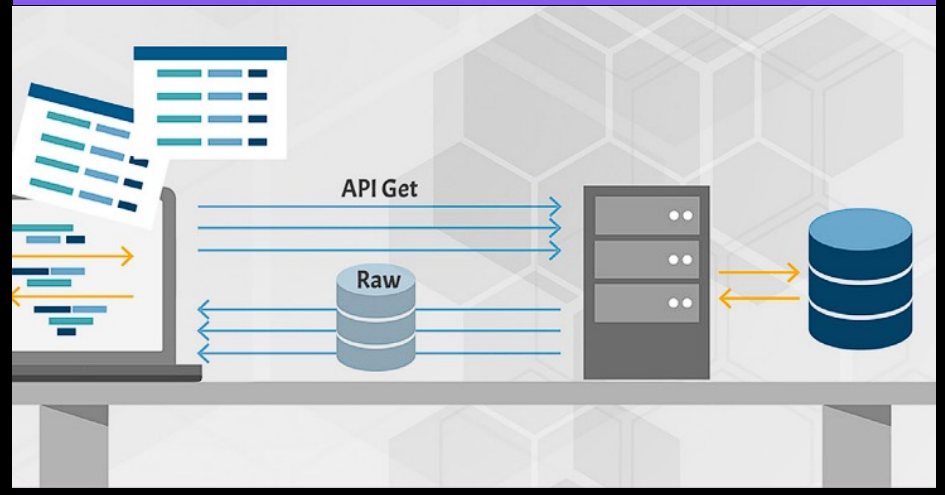
Attacker calls the API directly and gets sensitive data



https://www.youtube.com/watch?v=rr0f74YmvVQ



03 – Excessive Data Exposure



How to prevent

- Never rely on client, to filter data
- **Check the responses from the API to make sure they contain only legitimate data.**
- Backend developers should always ask themselves "who is the consumer of the data?" before exposing a new API endpoint.

- **Don’t forget about error responses!**

Headless & API date

SESSIONE TECH

API 06
Mass Assignment

Intesys.it

Use case

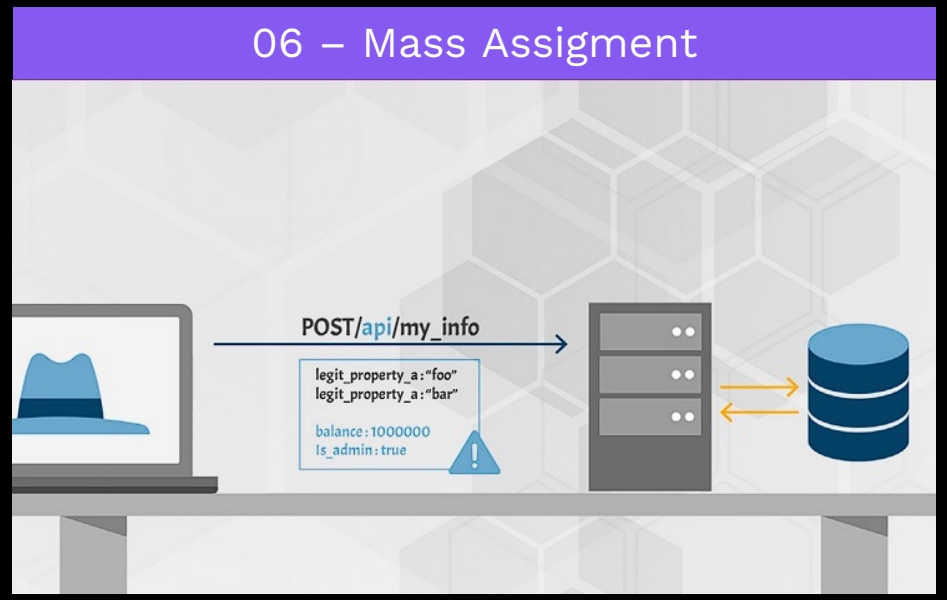
API is working with the data structures.

An attacker can update object properties that they should not have access to, allowing them to escalate privileges, tamper with data, and bypass security mechanisms

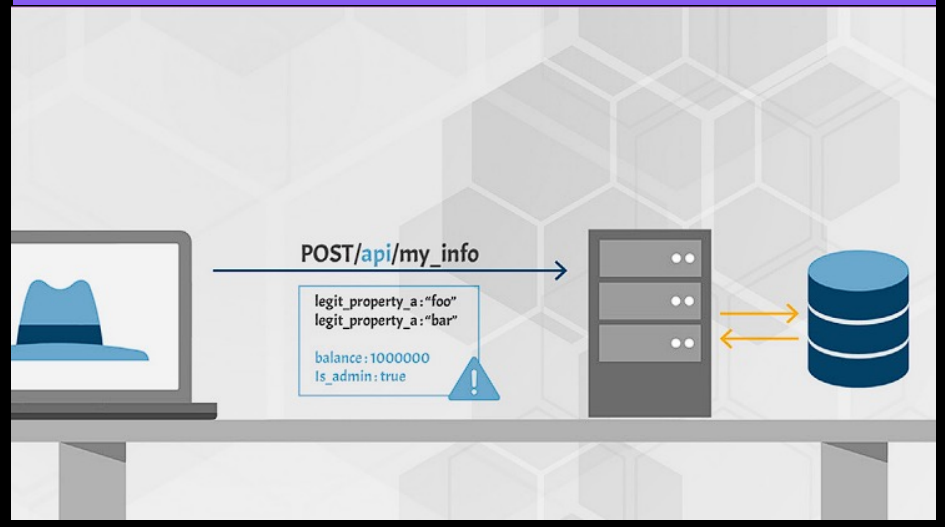
Received **payload is “blindly transformed into an object”** and stored

Legitimate - Client sends a legitimate request	**Attack** – Attackers sends the same request but adds the admin role in the request body

`PUT /api/v2/users/5deb9097 HTTP/1.1`
`{`
`"_id": "5deb9097",`
`"address": "******, NY City, NY",`
`"company_role": "Investment Services",`
`"email": "******",`
`"first_name": "******",`
`"full_name": "******",`
`"job_title": "Broker",`
`"last_name": "******",`
`"phone_number": "******"`
`}`
	`PUT /api/v2/users/5deb9097 HTTP/1.1`
	`{`
	`"_id": "5deb9097",`
	`"address": "******, NY City, NY",`
	`"company_role": "Investment Services",`
	`"email": "******",`
	`"first_name": "******",`
	`"full_name": "******",`
	`"is_admin": true,`
	`"is_sso": true,`
	`"job_title": "Broker",`
	`"last_name": "******",`
	`"permission_type": "admin",`
	`"phone_number": "******",`
	`"role": "admin",`
	`"sso_type": "admin",`
	`"system_user_type": "admin",`
	`"system_user_type_cd": 2,`
	`"user_type": "admin",`
	`"user_type_cd": 10`
	`}`



06 – Mass Assigment



How to prevent

- **Input validation!** Never rely on client
- Validate can happen on client side but **must be done always on server side**
- Whitelist the props allowed to the client

- If possible, **avoid** using functions that **automatically bind a client's input** into code **variables** or **internal objects**.

- **Use secure and updated serializer/deserializer libraries**

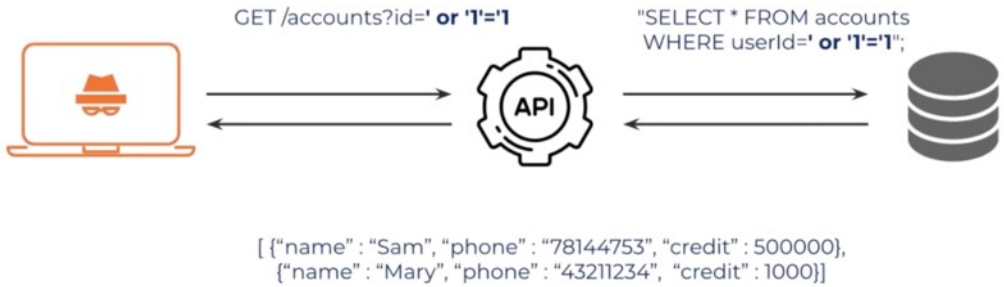
Test your API hammered with bad data

API 08
Injection

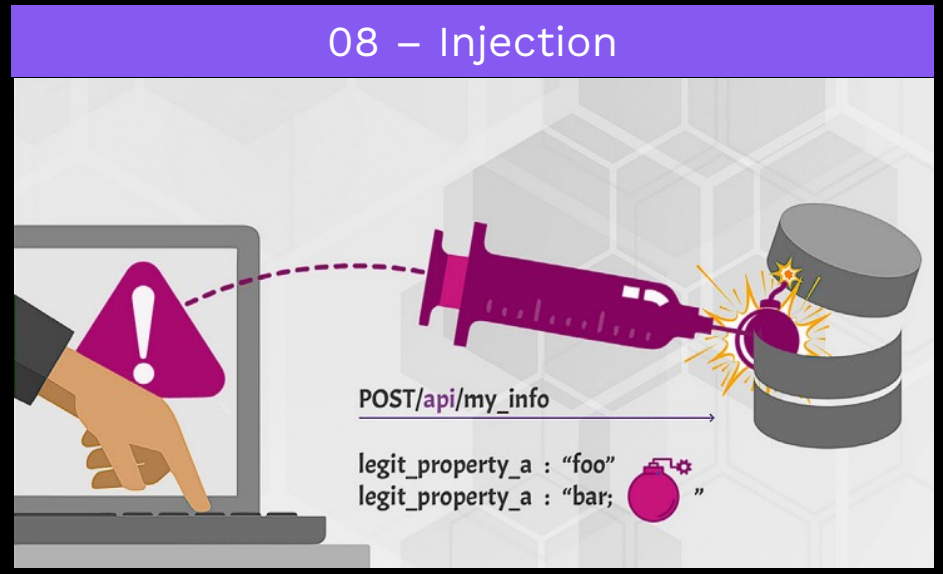
Use case

Attacker constructs API calls that include SQL-, NoSQL-, LDAP-, OS- and other commands that the API or **backend behind it blindly executes**

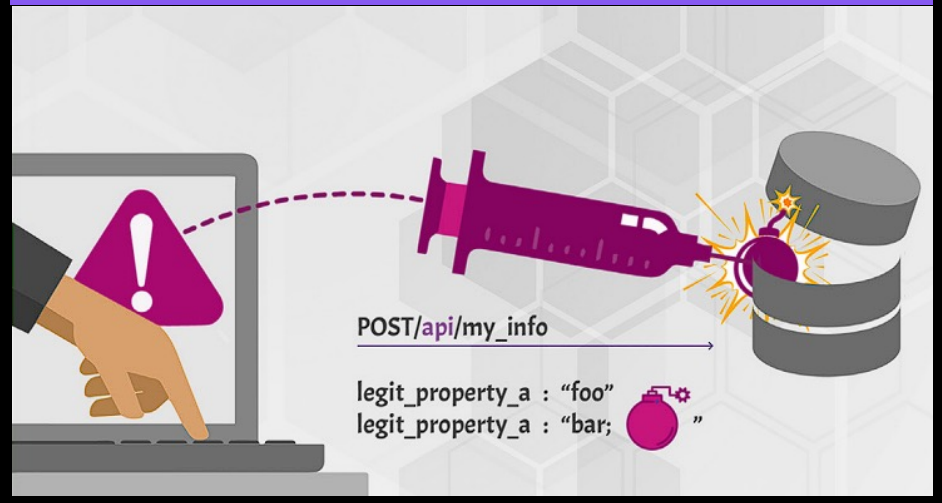
- SQL, NoSQL, LDAP, OS commands, ORM, ...



https://www.youtube.com/watch?v=rr0f74YmvVQ



08 – Injection



How to prevent

- **Input validation!** Never rely on client. **Validate, filter, sanitize all incoming data.**
    - Perform data validation using a single, trustworthy, and actively maintained library
- Define **data types** and **strict patterns for all string** parameters
- Define, limit, and enforce API outputs to prevent data leaks

Test this your API when hammered with bad data !

Headless & API date

SESSIONE TECH

Governance & Operations

Intesys.it

Headless & API date

SESSIONE TECH

API 07
Security Misconfiguration

Intesys.it

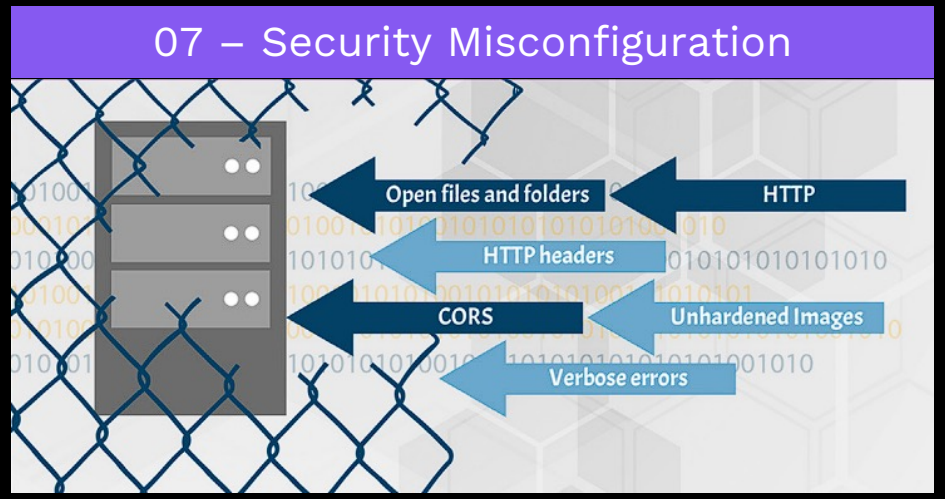
Use case

Poor configuration of the API servers allows attackers to exploit them.

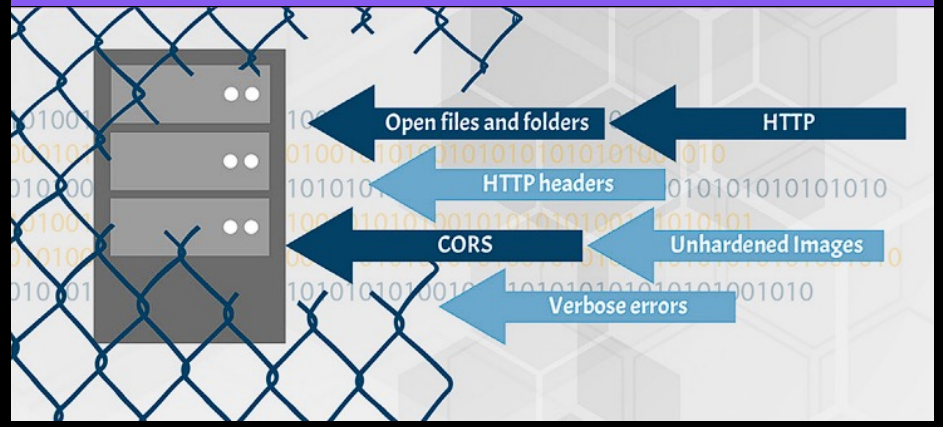
- misconfigured HTTP headers
- unnecessary features are enabled (e.g., HTTP verbs)
- permissive Cross-Origin resource sharing (CORS)
- verbose error messages
- the latest security patches are missing
- ...

66% of API incident are due for misconfigured APIs

IBM Security X-Force report - https://www.ibm.com/downloads/cas/WMDZOWK6



07 – Security Misconfiguration



How to prevent

- **Disable unnecessary features**
- **Automated process to locate configuration flaws**
- Repeatable hardening and patching processes
- Restrict administrative access

Headless & API date

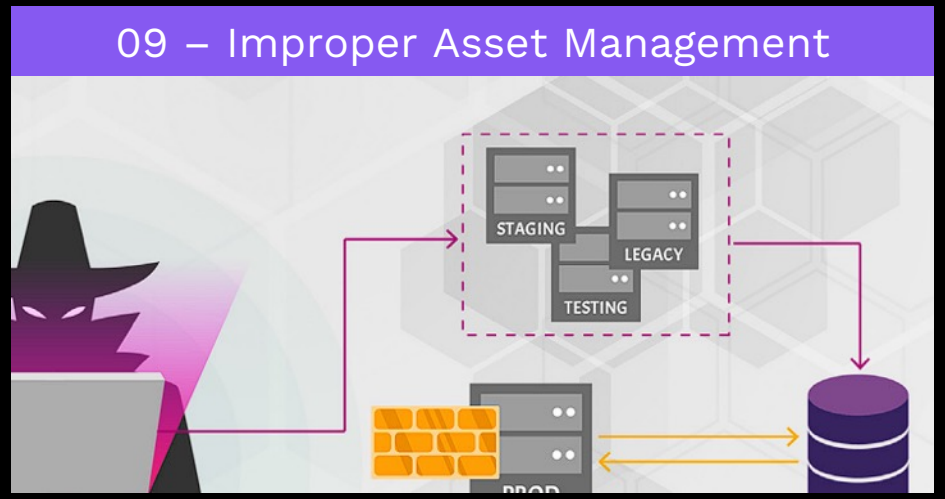
SESSIONE TECH

API 09
Improper Assets Management

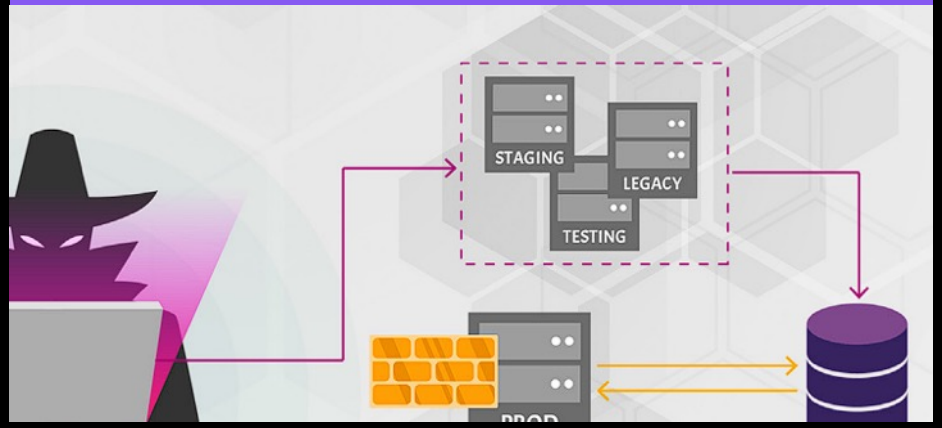
Intesys.it

Use case

Attacker finds **non-production versions of the API**: such as staging, testing, beta or earlier versions - **that are not as well protected**, and uses those to launch the attack



09 – Improper Asset Management



How to prevent

- **Inventory all API hosts**
- **Limit access to anything that should not be public**
- **Limit access to production data**: segregate access to production and non-production data.
- Implement additional controls such as API firewalls

- **Properly retire old versions or backport security fixes**

API 10

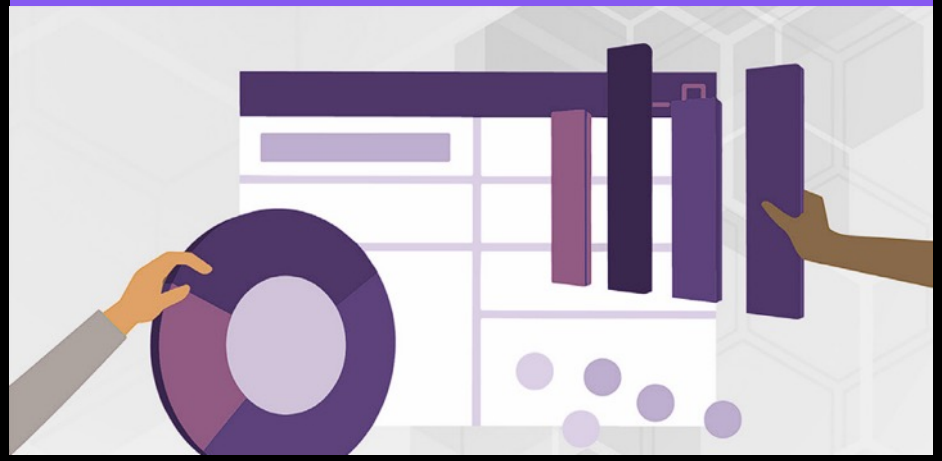
Insufficient Logging & Monitoring

Use case

Lack of proper logging, monitoring, and alerting let attacks go unnoticed

- Logs are not protected for integrity
- Logs are not integrated into SIEM (Security Information and Event Management systems)
- Logs and Alerts are poorly designed

10 – Insufficient Logging & Monitoring



How to prevent

- Include enough detail in your logs to identify attackers:
  - **login failed attempts**,
  - **denied access**,
  - ...
  - **input validation failures**, any **failures in security policy checks**
- **Avoid having sensitive data in logs** - If you need the information for debugging purposes, **redact it partially**.
- Ensure that logs are formatted to be consumable by other tools
- Integrate with SIEMs and other dashboards, monitoring, alerting tools

Headless & API date

SESSIONE TECH

Remarkable Actions For API SECURITY

Intesys.it

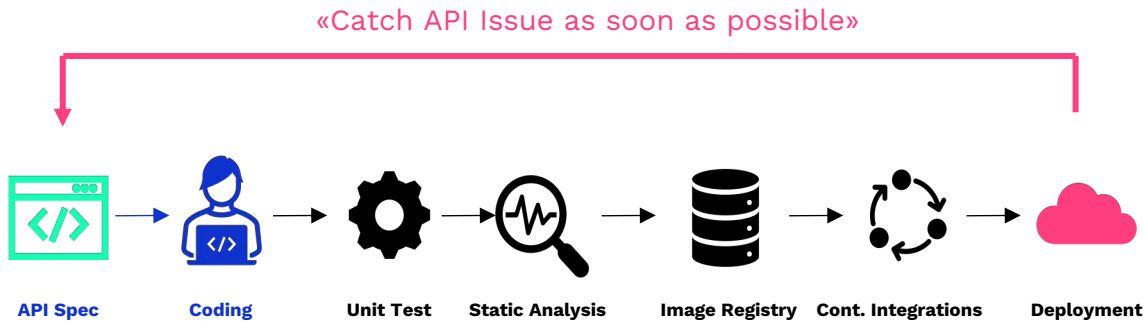
Remarkable Actions for API Security

1. **Utilizzare strumenti di API Security & API Management**
2. API First & Shift Left
    1. API Spec & Security Development Guidelines
3. Security Test, test, test!
4. ApiOps: GitOpts and DevSecOps: automate whole API Lifecycle

- **Firewalls & WAF (Web Application Firewall)**
- **API Gateway**
    - Traffic control & Security policies
    - Authentication and Authorization
    - Observability: reporting, alterting, log & tracing
- **API Access Management**
    - MFA (Multi Factor Authentication)
    - **SSO, OIDC** or **OAuth2** support /w IdP
    - Integration /w LDAP and secure identity repositories

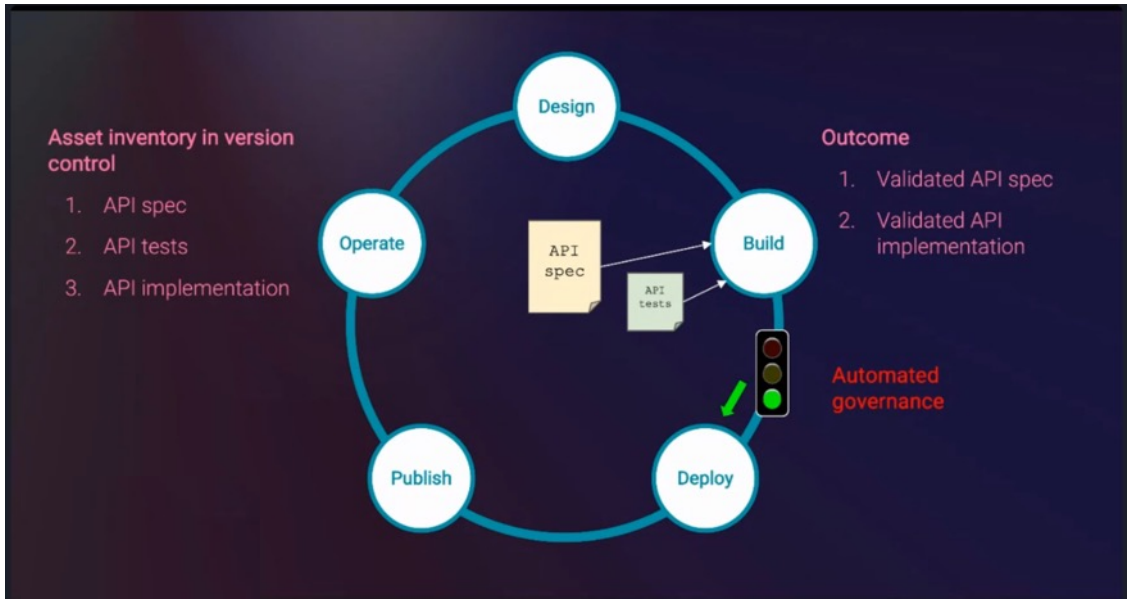
Remarkable Actions for API Security

1. Utilizzare strumenti di API Security & API Management
2. **API First & Shift Left**
    1. API Spec & Security Development Guidelines
3. Security Test, test, test!
4. ApiOps: GitOpts and DevSecOps: automate whole API Lifecycle



Remarkable Actions for API Security

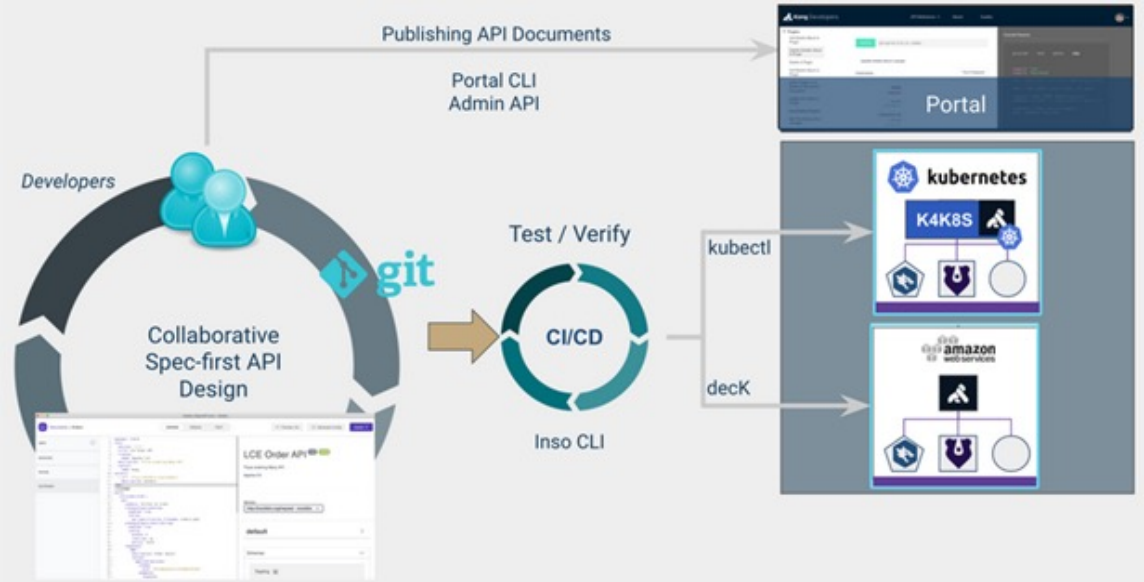
1. Utilizzare strumenti di API Security & API Management
2. API First & Shift Left
    1. API Spec & Security Development Guidelines
3. **Security Test, test, test!**
4. ApiOps: GitOpts and DevSecOps: automate whole API Lifecycle



https://youtu.be/JI_rmIxMSVk

Remarkable Actions for API Security

1. Utilizzare strumenti di API Security & API Management
2. API First & Shift Left
    1. API Spec & Security Development Guidelines
3. Security Test, test, test!
4. **ApiOps: GitOpts and DevSecOps: automate whole API Lifecycle**
5. **Collaboration between: API Team, Dev Team, Ops and Security Team**



Headless & API date

Liferay | mia Platform | Intesys

Grazie

Denis Signoretto

IT ARCHITECT, INTESYS

denis.signoretto@intesys.it