Service Exposure Challenges in APIs

Niranth Amogh
Head - 6G Ecosystem & Standardization
Nokia R&D India, Bengaluru

10th Aug 2023

INTERNATIONAL CONFERENCE ON 5G NETWORK SECURITY
ORGANIZED BY: NCCS, DoT, C-DoT
VENUE: B.M.S COLLEGE OF ENGINEERING, BENGALURU

AGENDA

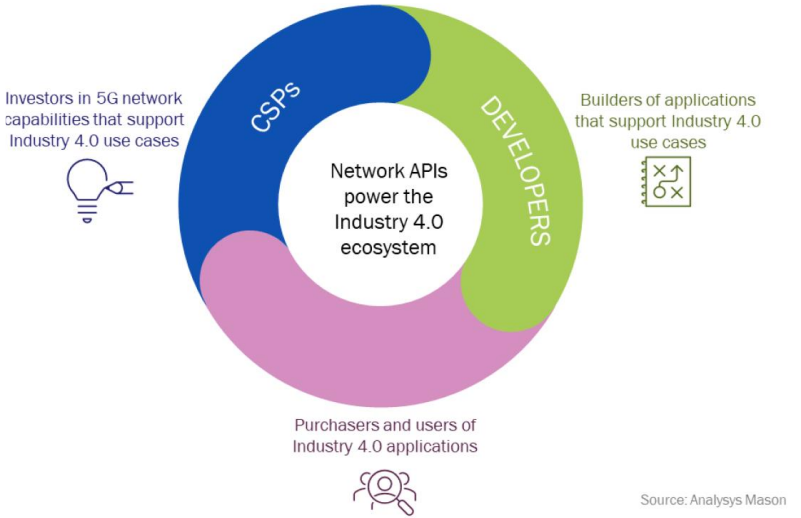
- Context
- OWASP API Security Top 10 2023
- API Exposure Framework Standard

Context

Demand-Supply Overview

Envisioning huge benefits in the Ecosystem around Telecom APIs

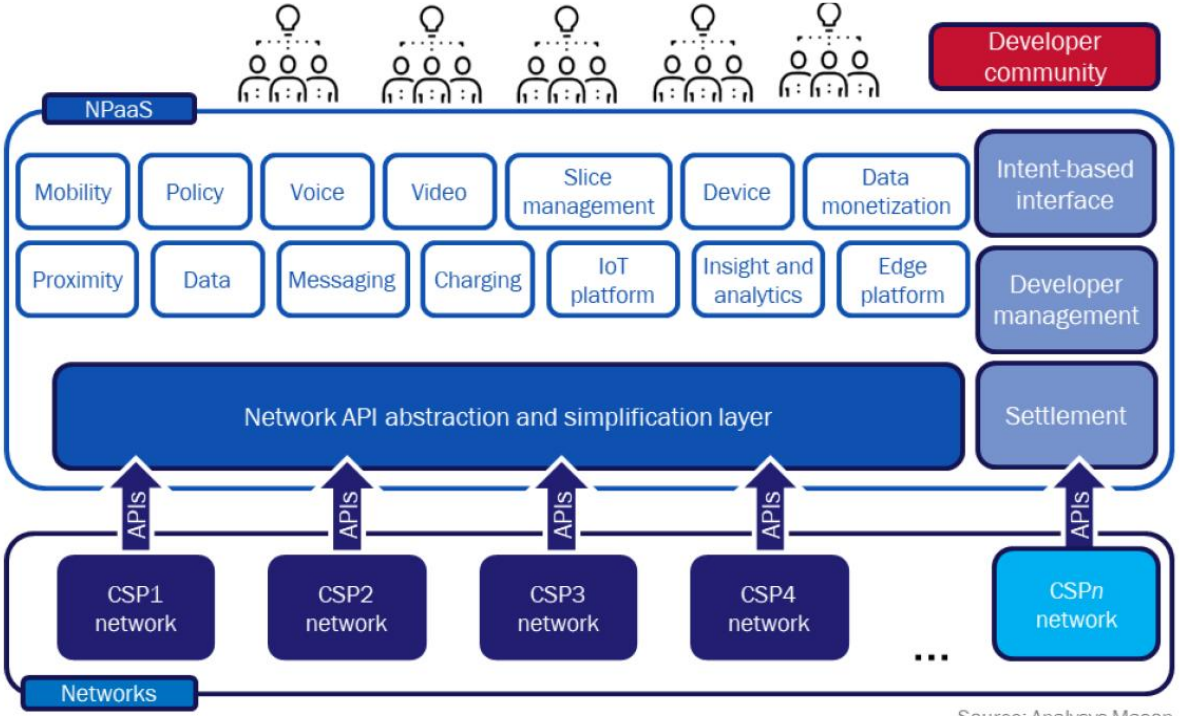
- Communications service providers (CSPs) invested USD103 billion in building 5G networks in 2022 and are expected to spend a further USD129 billion in 2023.
  - A recent study[1] reveals that the vast majority of communication service providers (CSPs) – **73% – view network API exposure as a top 5 priority**.
- Developers are not network specialists so they need a familiar way of accessing and incorporating a wide range of network capabilities into their applications, through simplified APIs, software development toolkits (SDKs) and code development support services.
  - **60% of developers are interested in writing software programs specifically to enable them to enrich applications with new services**, such as improved quality on demand.
- Between 2022 and 2026, the **market for services enabled by telecom APIs is forecast to grow from $12 billion to $34 billion**.



[1] https://onestore.nokia.com/asset/i/213081?_ga=2.239017280.1130352326.1681860459-13826807.1559934403&_gac=1.184471764.1680103771.Cj0KCQjww4-hBhCtARIsAC9gR3buJglAoKxfvESoz8ims2ruBI7NjqEPeT8n4WrWlLe6T2zFQnjvKKsaAq1XEALw_wcB

NPaaS

CSPs strategy to monetize from 5G and beyond



- 76% of software developers believe that network APIs should be user-friendly and **secure**.
- CSPs must ensure their network APIs are accessible, user-friendly, **secure** and accompanied by comprehensive documentation to encourage developer adoption and enable successful 5G service delivery.
- NPaaS delivers:
  - Faster TTM
  - RESTful APIs (Integration ease)
  - Security at every level
  - Performance and Scale (Leveraging Cloud and Edge)

Source: Analysys Mason

Perspectives on Challenges

- **Level of Exposure**
  - Exposure of sensitive information.
  - Achieving right level of abstraction.
  - Business requirements driving APIs to change frequently.
- **Security**
  - Threat Protection - API attacks are increasing. Majority of API attack attempts can be mapped to OWASP API Security Top 10 methods.
  - Access Control – Authentication, Authorization, Accounting
  - API security sufficiency impacts business roll out decisions.
  - One size doesn’t fit all. (Common aspects and Specific aspects)
- **Standardization**
  - Lack of API standards or Lack of awareness of API standards.

OWASP API Security Top 10 2023[1]

- **API1:2023 - Broken Object Level Authorization**
- **API2:2023 - Broken Authentication**
- **API3:2023 - Broken Object Property Level Authorization**
- **API4:2023 - Unrestricted Resource Consumption**
- **API5:2023 - Broken Function Level Authorization**
- **API6:2023 - Unrestricted Access to Sensitive Business Flows**
- **API7:2023 - Server Side Request Forgery**
- **API8:2023 - Security Misconfiguration**
- **API9:2023 - Improper Inventory Management**
- **API10:2023 - Unsafe Consumption of APIs**

[1] https://owasp.org/www-project-api-security/

API1:2023 Broken Object Level Authorization (BOLA)

Description

- Attackers can exploit API endpoints that are vulnerable to broken object-level authorization by manipulating the ID of an object that is sent within the request.
- 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.
- This attack is also known as IDOR (Insecure Direct Object Reference).

Use Cases

- API call parameters use IDs of resourced accessed by the API
  - /api/user1/details
- Attackers replace the IDs of their resources with a different, which they guessed
  - /api/user2/details
- The API does not check permissions and lets the call through
- Problem is aggravated if IDs can be enumerated
  - /api/123/details

How to Prevent

- Implement authorization checks with user policies and hierarchy
- Don’t rely on IDs sent from client. Use IDs stored in the session object instead.
- Check authorization each time there is a client request to access database
- Use random non-guessable IDs (UUIDs)

API2:2023 Broken Authentication

Description

- Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently. Compromising a system's ability to identify the client/user, compromises API security overall.
- Poorly implemented API authentication allowing attackers to assume other users' identities.

Use Cases

- Unprotected APIs that are considered "internal"
- Weak authentication not following industry best practices
- Weak, not rotating API keys
- Weak, plain text, encrypted, poorly hashed, shared/default passwords
- Susceptible to brute force attacks and credential stuffing
- Credentials and keys in URL
- Lack of access token validation (including JWT validation)
- Unsigned, weakly signed, non-expiring JWTs (JSON Web Token)

How to Prevent

- Check all possible ways to authenticate to all APIs
- Password reset APIs and one-time links also allow users to get authenticated and should be protected just as seriously
- Use standard authentication, token generation, password storage, MFA
- Use short-lived access tokens
- Authenticate your apps (so you know who is talking to you)
- Use stricter rate-limiting for authentication, implement lockout policies and weak password checks

API3:2023 Broken Object Property Level Authorization

Description

- This category combines API3:2019 Excessive Data Exposure and API6:2019 - Mass Assignment, focusing on the root cause: the lack of or improper authorization validation at the object property level.
- This leads to information exposure or manipulation by unauthorized parties.

Use Cases

- APIs return full data objects as they are stored by the database
- Client application shows only the data that user needs to see
- Attacker calls the API directly and gets sensitive data
- API working with the data structures
- Received payload is blindly transformed into an object and stored
- Attackers can guess the fields by looking at the GET request data

How to Prevent

- Never rely on client to filter data
- Review all responses and adapt responses to what the API consumers really need
- Define schemas of all the API responses
- Don’t forget about error responses
- Identify all the sensitive or PII info and justify its use
- Enforce response checks to prevent accidental data and exception leaks
- Don’t automatically bind incoming data and internal objects. Explicitly define all the parameters and payloads you are expecting
- For object schemas, use the readOnly set to true for all properties that can be retrieved via APIs but should never be modified
- Precisely define at design time the schemas, types, patterns you will accept in requests and enforce them at runtime.

API4:2023 Unrestricted Resource Consumption

Description

- API is not protected against an excessive amount of calls or payload sizes. Attackers use that for DoS and brute force attacks.

Use Cases

- Attacker overloading the API
- Excessive rate of requests
- Request or field sizes
- “Zip bombs”

How to Prevent

- Rate limiting
- Payload size limits
- Rate limits specific to API methods, clients, addresses
- Checks on compression ratios
- Limits on container resources

API5:2023 Broken Function Level Authorization

Description

- API relies on client to use user level or admin level APIs.
- Attacker figures out the “hidden” admin API methods and invokes them directly.

Use Cases

- Some administrative functions are exposed as APIs
- Non-privileged users can access these functions if they know how
- Can be a matter of knowing the URL, using a different verb or parameter
  - /api/users/v1/user/myinfo
  - /api/admins/v1/users/all

How to Prevent

- Don’t rely on app to enforce admin access
- Deny all access by default
- Grant access based on specific roles
- Properly design and test authorization

API6:2023 Unrestricted Access to Sensitive Business Flows

Description

- Exploitation usually involves understanding the business model backed by the API, finding sensitive business flows, and automating access to these flows, causing harm to the business.
- APIs vulnerable to this risk expose a business flow - such as buying a ticket, or posting a comment - without compensating for how the functionality could harm the business if used excessively in an automated manner.

Use Cases

- E2E business flows are exposed without adequate checkpoints.
- Making excessive requests for high priority QoS flows.

How to Prevent

- Business - identify the business flows that might harm the business if they are excessively used.
- Engineering - choose the right protection mechanisms to mitigate the business risk.
- Deny service for unexpected users/UEs
- Secure and limit access to APIs that are consumed directly by machines (such as developer and B2B APIs)

API7:2023 Server Side Request Forgery

Description

- Server-Side Request Forgery (SSRF) flaws can occur when an API is fetching a remote resource without validating the user-supplied URI.
- This enables an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall or a VPN.

Use Cases

- API use Application supplied URI to fetch data.
  - POST /api/users/u1 {URL: XXX}
- Callback URL modification

How to Prevent

- Isolate the resource fetching mechanism in your network: usually these features are aimed to retrieve remote resources and not internal ones.
- Enable only allowed list of URLs – Make it specific or verifiable
- Parse URLs and verify
- Do not send raw data in responses.

API8:2023 Security Misconfiguration

Description

- Poor configuration of the API servers allows attackers to exploit them.
- APIs and the systems supporting them typically contain complex configurations, meant to make the APIs more customizable. Software and DevOps engineers can miss these configurations, or don't follow security best practices when it comes to configuration, opening the door for different types of attacks.

Use Cases

- Unpatched systems
- Unprotected files and directories
- Unhardened images
- Missing, outdated, misconfigured TLS
- Exposed storage or server management panels
- Missing CORS (Cross Origin Resource Sharing) policy or security headers
- Error messages with stack traces

How to Prevent

- Repeatable hardening and patching processes
- Automated process to locate configuration flaws
- Disable unnecessary features
- Restrict administrative access
- Define and enforce all outputs including errors

API9:2023 Improper Inventory Management

Description

- APIs tend to expose more endpoints, making proper and updated documentation highly important.
- A proper inventory of hosts and deployed API versions also are important to mitigate issues such as deprecated API versions and exposed debug endpoints.
- 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.

Use Cases

- DevOps, cloud, containers, K8S make having multiple deployments easy (Dev, Test, Branches, Staging, Old versions)
- Desire to maintain backward compatibility forces to leave old APIs running
- Old or non-production versions are not properly maintained
- These endpoints still have access to production data
- Once authenticated with one endpoint, attacker may switch to the other

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 external controls such as API firewalls
- Properly retire old versions or backport security fixes
- Implement strict authentication, redirects, CORS, etc.

API10:2023 - Unsafe Consumption of APIs

Description

- Developers tend to trust data received from third-party APIs more than user input, and so tend to adopt weaker security standards. In order to compromise APIs, attackers go after integrated third-party services instead of trying to compromise the target API directly.
- Attackers identify and potentially compromise other APIs/services the target API integrated with

Use Cases

- Mashup services – Combining data from multiple 3rd parties
- Cascaded API invocations: One API invokes another API, the other API invokes yet another API.

How to Prevent

- Identify the API invoker and employ adequate security. (TLS)
- Define API responses clearly and only prepare API responses by extracting the necessary data from the data obtained from multiple services.
- Identify loopholes which lead to cyclic cascades or infinite cascades.

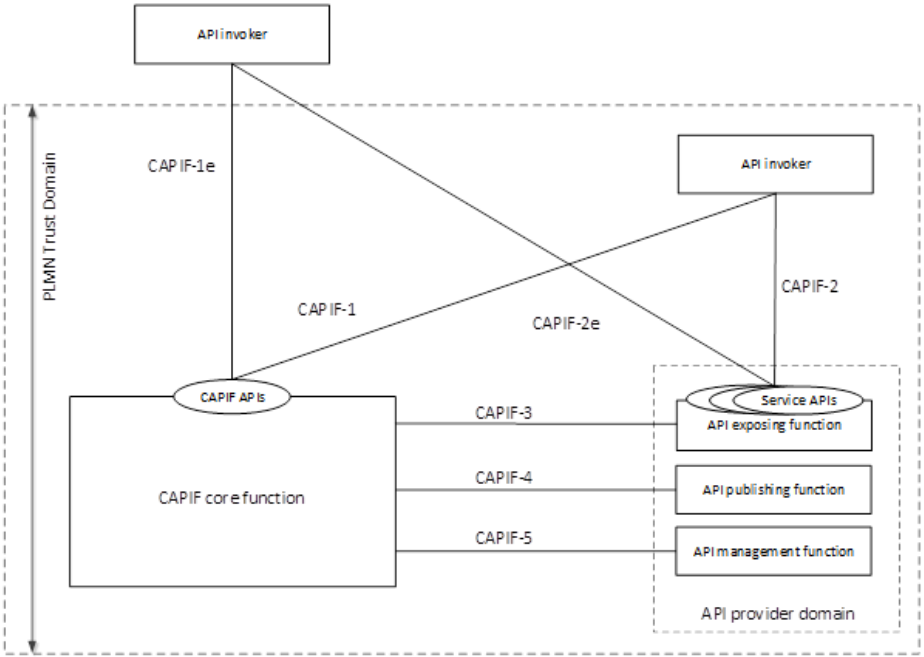
API Exposure Framework

CAPIF – Common API Framework by 3GPP

3GPP TS 23.222 - https://www.3gpp.org/DynaReport/23222.htm

CAPIF Architecture

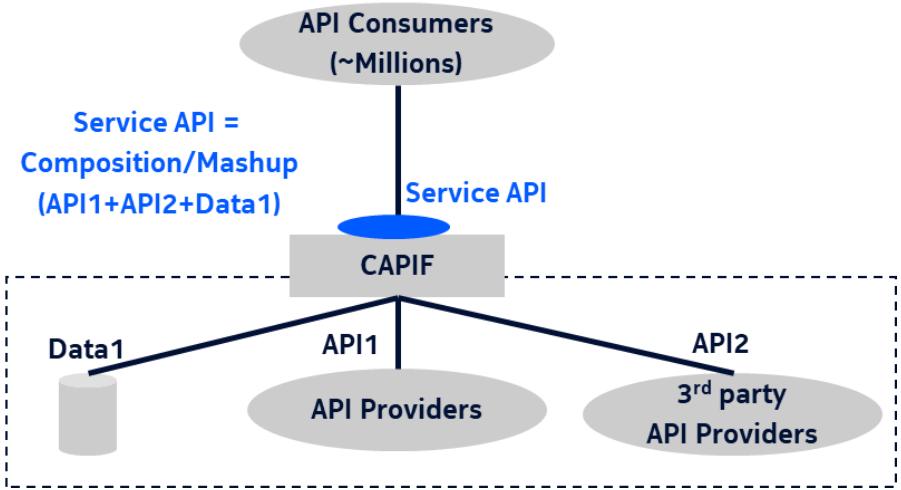
3GPP standard for the exposure and consumption of APIs in 5G networks

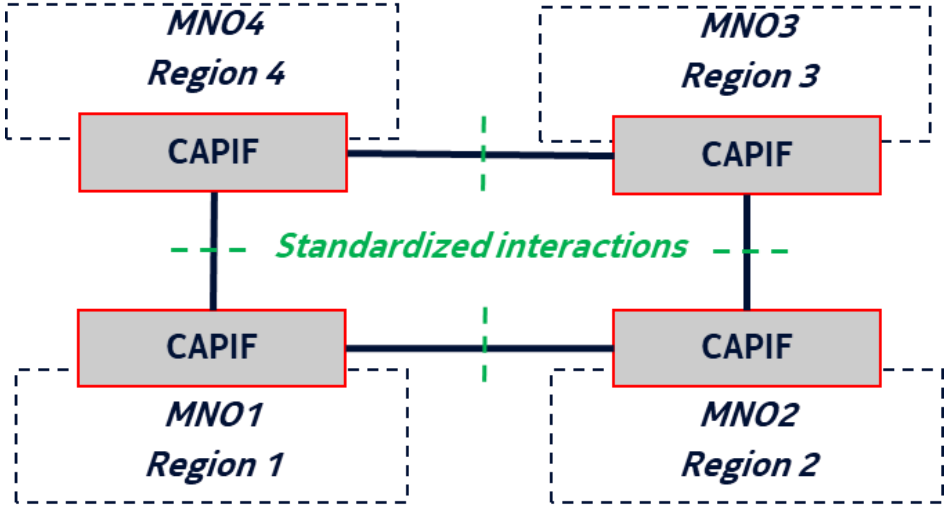


- **API Invoker**: Consumer of APIs (e.g. Applications in Network or even on the UEs)
- **API Provider**: Provider of APIs.
  - **AEF (API Exposing Function)** – Enables API invocation.
  - **APF (API Publishing Function)** – Publishes the API.
  - **AMF (API Mgmt Function)** – Mgmt of API lifecycle, performance, etc.
- **CCF (CAPIF Core Function)**: Common set of services to enable secure interactions for API consumptions.
  - Publish, Discover, Events (API related, API invoker related), API invoker onboarding, Authentication and Authorization for API invocations, Access Control (also supporting Cascaded API invocations), API Routing, Logging/Auditing, Charging, API Topology Hiding,

CAPIF Architecture – Support for 3rd Parties

Enabling secure API driven ecosystem





CAPIF - API consistency guidelines

Developing Requirements for API specifications

Fundamental API Guidelines

- the function of the API;
- the resource(s) or endpoints involved;
- the list of supported operations and their usage;
- the list of input and output parameters along with applicable schemas, as required;
- the list of supported response codes;
- the behaviour of the network entity exposing the APIs (e.g. the CAPIF core function or the API exposing function) for each supported operation;
- the list of applicable data types; and
- the list of applicable protocols and data serialization formats.

API Design and Architecture

- Uniform interface with resource/data model with
  - implementation of the resource(s)/operations involved in the APIs should be hidden from the client.
  - Support adequate operation on the resources.
- Independent evolution of Client and Server.
- Stateless, support CRUD with appropriate response codes
- Support for cacheability (client or server)
- Security
- Versioning

Summary

- APIs will play key role in monetization of 5G and beyond deployments for the CSPs.
- API security is of utmost importance
  - Applicable to every stakeholder in the API supply chain
- A standardized API Exposure Framework like CAPIF enables the consistent definition of secure APIs.

Thank You!
At Nokia, we create technology that helps the world act together