International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V68I10P103

Volume 68 Issue 10, 14-20, October 2020
© 2020 Seventh Sense Research Group®

Original Article

From Batter to Cake: Bake your Own Security Model in API Management

BharathanKasthuriRengan

Principal Architect, Virtusa Corporation,65 W Newell Ave, Rutherford, NJ, USA.

Received Date: 13 September 2020
Revised Date: 16 October 2020
Accepted Date: 18 October 2020

Abstract *- APIs' growth originates from an elementary need for a better way to encapsulate and share information and enable transaction processing between elements in the solution stack. Unfortunately, APIs have often been treated as tactical assets until relatively recently. The idea behind APIs has existed since the beginning of computing; however, in the last 10 years, they have grown significantly in number and sophistication.*
They are increasingly scalable, monetized, and ubiquitous, with more than 12,000 listed on the Web, managing a global API directory.
Defining API security is more than a strategy, as it has to have an immediate impact through the tactical solution. Defining a key security framework extending the API Management platform's vendor products is key to API adoption success.
This whitepaper covers the security framework guidelines, reference implementation (end to end from API development to deployment and governance) for a large enterprise.

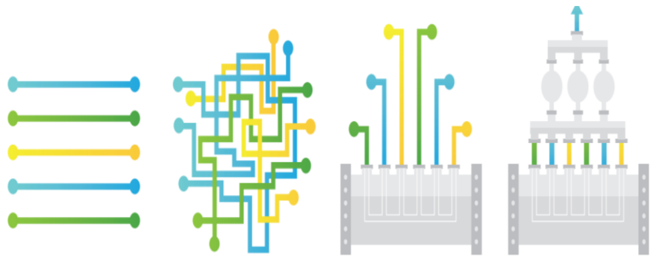
Keywords *- API Security, Custom Security, API Management, API Governance, API Gateway*

I. INTRODUCTION

The idea behind APIs has existed since the beginning of computing; however, in the last 10 years, they have grown significantly in number and sophistication. They are increasingly scalable, monetized, and ubiquitous, with more than 12,000 listed on ProgrammableWeb, which manages a global API directory.
Future-looking scenarios involving smartphones, tablets, social outlets, wearables, embedded sensors, and connected devices will have inherent internal and external dependencies on underlying data and services. APIs can add features, reach, and context to new products and services or become products and services themselves.

A. Evolution of APIs

The idea behind APIs has existed since the beginning of computing; however, in the last 10 years, they have grown significantly in number and sophistication. They are increasingly scalable, monetized, and ubiquitous, with more than 12,000 listed on ProgrammableWeb, which manages a global API directory.



Source:http://www.programmableweb.com accessed January 7, 2015.

1960–1980	**1980–1990**	**1990–2000**	**2000-Today**
Basic interoperability enables the first programmatic exchanges of information	Creation of interfaces with function and logic.Object brokers, procedure calls, and program calls	New platforms enhance exchanges through middleware. Interfaces begin to be defined as services.	Businesses build APIs to enable and accelerate new service development and offerings.
Techniques ARPANET, ATTP, and TCP sessions	**Techniques** Point-to-point interfaces, screen scraping, RFCs, and EDI.	**Techniques** Enterprise service bus and service-oriented architecture	**Techniques** Integration as a service, RESTful services, API management

B. Three Major Pillars-API Adoption

Given the future of API and its impact on the economy, investing in an API management platform is critical to any enterprise's success.
(3) Three important pillars are crucial for the success of API adoption.

- Create, govern, and deploy APIs: versioning, discoverability, and clarity of scope and purpose
- Secure, monitor, and optimize usage: access control, security policy enforcement, routing, caching, throttling (rate limits and quotas), instrumentation, and analytics
- Market, support, and monetize assets: manage sales, pricing, metering, billing, and key or token provisioning

C. Critical Pillar- Security

API security should be an integral part of an API implementation, and achieving this requires a specific view of architecture.

API gateways allow developers to encapsulate an application's internal structure in multiple ways depending upon the use case. In addition to accommodating direct requests, gateways can invoke multiple back-end services and aggregate the results.

Following are the critical focus areas of API security

- Enhance API lifecycle management, including publishing, Monitoring, protecting, analyzing, monetizing, and engaging the community.
- Protect APIs from network threats, including denial-of-service (DoS) attacks and common scripting/injection attacks through a web application firewall (WAF)
- Protect data from being aggressively scraped by detecting patterns from one or more IP addresses through anti-farming/bot security
- Distribute cached content to the edge of the Internet,
- Manage identity, authentication, and authorization services, often through integration with API gateway and management layers via Identity Providers (IdP)
- Perform thorough security assessment for existing and new build APIs to identify vulnerabilities before release across technical and business aspects. API security assessments consistently use globally accepted and industry-standard frameworks.

API security architecture (in figure – 2) illustrates the components and the layers of security.

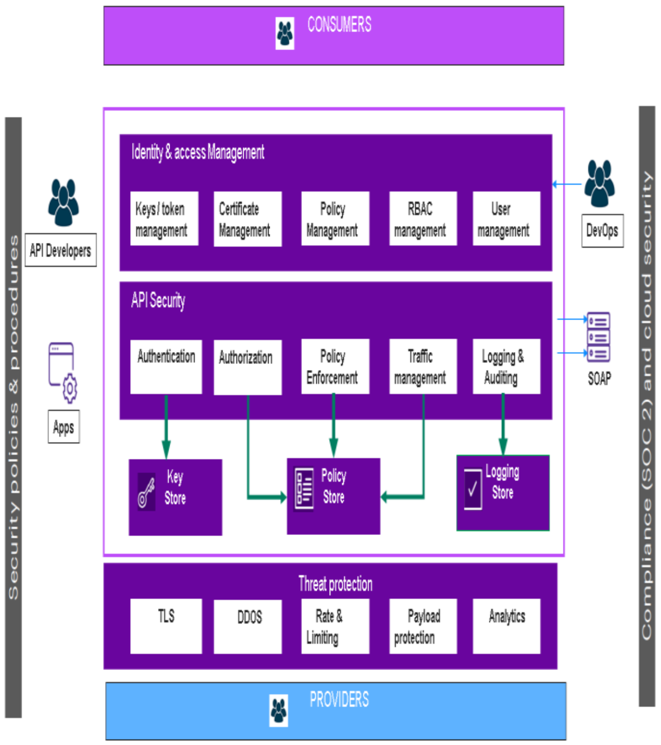


Fig. 2 – API Security (Component) Architecture

D. Custom Security for APIs in Nutshell

For large enterprises, API security vendor solutions are not an exact fit for their risk exposure as they differ subtly across the organization.

The risks associated are different across the business portfolio, even within the Enterprise.

It is not like "one size fits all," as they bind heterogeneous solutions. Some of the API security can span over networks, layers, data, and applications. The industry is governed by a global regulatory body that enforces what kind of data, who has access, what was done, information to be shared and tracked (audit trail).

This forces enterprises to think beyond vendors' traditional solutions, track end-to-end transaction lifecycle, audit reports, mask sensitive information, and log retention of 10+ years of transactions.

Considering the situation, it is worth considering building one's custom policy, Monitoring, and management layers over and beyond vendors. Some custom security policies or implementations are highlighted below:

E.API Development

- Identity Management (IM) in conjunction with API Management
- Deep entitlements for critical operations in transaction management (Admin functions, supervisory roles at the store, to override standard functions)
- API portal functions (to publish and subscribe APIs with an integrated workflow for customer-facing APIs) management, control panes of API management

- API gateway security policy customizations (OAuth scopes with custom grant type handler over and beyond general grant types, hybrid scenarios (On-Prem and On Cloud functions where there can be providers and consumers on both ends) – **data panes**
- Masking of CII data, encryption of data (sensitive data on APIs) at run time – purely dynamic considering the nature of business – **Layer 7 security**
- DAST and SAST – dynamic security testing of APIs and applications as part of the development lifecycle.

F. API Governance

Active Monitoring (configured alerts) of user management provisioning of users on demand by privileged users

- Successful and failure login (by privileged and normal users) in API portal, Admin portals (Provisioning of users (non-admin) by admin users in APIM context which could be, organization-wide roles, intra organization roles, deputy organization admin roles, the entitlement carefully crafted by the super admin
- User to roles and resource management with a detailed report on usage (dynamic) specific to API
- Log management (clean up, audit log malware, etc.) of Identity providers, APIM
- Event management (import /export of users, role management, dynamic client registration, secret management) on APIM
- Secure certificate management - API management and back end services
- Master configurations of API management (proxy, identity roles, user, LDAP configuration).

G. API Promotion

Promotion of API from the development environment to production (where external partners consume it

In the next few sections of the whitepaper, we are going to see how do we implement the above (a-f scenarios) from defining policies, implementing them, governing them (through Monitoring, enforcement, alerts) at the API management platform (API portal, API policy manager, API Gateway layers).

This white paper is focused on building your multi-layer security model that provides advanced security for enterprise APIs.

H. Extensions to API Gateway Security

Along with the Identity Access Management platform, Enterprise can define OAuth scopes for an individual resource inside API (as security definition), thereby providing an additional security layer.

Define custom grant type on Token Provider endpoint to implement custom security solution so that highly secured APIs, only applications registered with this custom grant type, are provided access to these APIS.

Custom OAuth token provider with a custom solution that generates OAuth token (with OAuth scopes as eligible claims) as an exchange for JWT token (with requested scopes) from the Identity Access Management solution.

I. Extensions to API Publisher Security

- API Publisher portal provides individual API designers with the ability to work and update Swagger /Open API with multiple endpoints, define the models and payload. However, this may not align with enterprise guidelines. Designers submit the new API definition or a new API version to the organization to address this. Every department within the organization can have its workflow, including its security policy. For the New API version, there can be 1 level of approval. There can be 0 or 1 approval to edit existing API based on major, minor, or revision of API.
- b. API administrators (for every sub-organization) review new API definitions, API versions and either approve or deny API.
- c. Customize security policy so that every sub organization can leverage and share client credentials instead of user-level credentials.
- d. Create tenants for every department in an organization. Set new policies for every department in an organization with a specific role call out like publisher, import/export, subscription, etc.
- e. Define Grant types, Token expiration time, JWT/OAuth token type at either application/user level for every Tenant.
- f. For any API to be crossly subscribed by other departments, it would involve additional API owners' approval.
- g. Auto subscription of applications (owner department) to its APIs.
- h. Create user segmentation, approve API organization, API, versions, and user access to provide specific entitlements.

J.Extensions to API Policy Manager Security

- Provide rate limiting and throttling (Traffic Manager) on JWT, OAuth token claims. Rate limits can be provided based on individual user, department, role.
- Provide application level throttling in addition to the API level. Every department can be provided specific API access (like 1000 requests per day to 5000 requests per day based on department).

K. Masking and Encryption of data at Transit, ReST- API Security

- Define sensitive data at API definition, at the implementation level (Service Provider), and provide masking requirements wherever possible.

- Other sensitive information can be scattered around logs, database end. This can be encrypted and can be decrypted for further processing.

L. API Security testing with SAST, DAST

Integrate with CI/CD pipeline, check for SAST, DAST testing during APIs onboard, and back end services. Check on back end services dependencies, root-level access, API metadata storage in Persistent Volume, API secrets management with PaaS

M. Implementation

Extend the API policies and API portal capabilities to provide coarse-grained, fine-grained API restricted access to critical resources. This can be an additional layer of security.

a) Custom JWT Handler

Enterprise can define OAuth scopes for an individual resource inside API (as security definition), thereby providing an additional layer of security

- New custom grant types can be mapped to the API gateway to build our solution. Following are the allowed grant types in a typical OAuth provider
    1. Authorization code
    2. Refresh Token
    3. Password
    4. Client credentials
    5. JWT

This new grant type can be built as an exchange of tokens (JWT) from IAM, with eligible OAuth scopes specific to each user to return OAuth 2 tokens.
The API gateway exposes a token provider endpoint with a custom grant type to return OAuth 2 tokens.

- Only specific API Management Applications can access these custom grant types. We can configure specific applications (client credentials) to these grant types.
- API gateway provides extensive support on advanced traffic policies. Most of the traffic management policies are out of the box. However, we can build customized solutions such as fine-grained policies to allow requests based on

JWT claims -> throttle limits based on claim name, value. E.g., we want to allow user claims which have a department matching specific values, let's say IT, which can be granted 2500 requests/hour. However, we want to allow traffic to APIs for a department like the front office, 500 requests/hour. Such advanced throttling policies.
For internal staff, administrators, we want to provide privileged access to 3000 requests/hour (or higher as an example). This can be based on an additional service token on the request header specific to each backend service and API. These service tokens can be added on demand by API gateway based on granular entitlements.

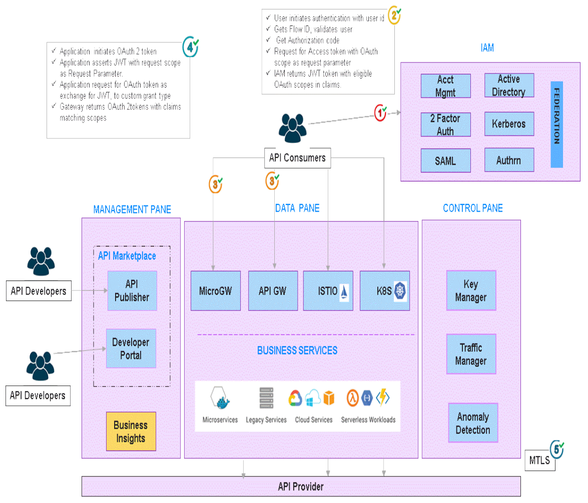
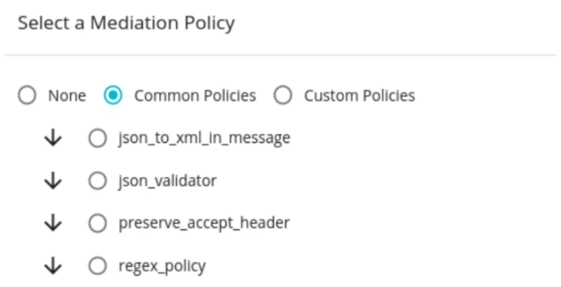

✓ Application initiates OAuth 2 token
✓ Application asserts JWT with request scope as Request Parameter.
✓ Application request for OAuth token as exchange for JWT, to custom grant type
✓ Gateway returns OAuth 2tokens with claims matching scopes
4
✓ User initiates authentication with user id
✓ Gets Flow ID, validates user
✓ Get Authorization code
✓ Request for Access token with OAuth scope as request parameter
✓ IAM returns JWT token with eligible OAuth scopes in claims.
2
IAM
Acct Mgmt
Active Directory
2 Factor Auth
Kerberos
SAML
Authrn
FEDERATION
1
API Consumers
3
3
MANAGEMENT PANE
DATA PANE
CONTROL PANE
API Developers
API Marketplace
API Publisher
Developer Portal
API Developers
Business Insights
MicroGW
API GW
ISTIO
K8S
BUSINESS SERVICES
Microservices
Legacy Services
Cloud Services
Serverless Workloads
Key Manager
Traffic Manager
Anomaly Detection
MTLS
5
API Provider

Fig. 3 Grant type

b) Mediation Policy Support

We can build custom mediation rules for inbound and outbound traffic from/to the gateway. We can add the default mediation policies to add specific ones suitable to each one, and if we want to propagate the header to back end services, it can be done by changing the mediation policies (refer to Fig.4)


Select a Mediation Policy

None [✓ Common Policies] Custom Policies

- json_to_xml_in_message
- json_validator
- preserve_accept_header
- regex_policy

Fig. 4 Mediation Policy

- Back end services are routed through API Gateway; however, API consumers can pass on Mutual TLS cert to API gateway
    1. API manager to have the public key of the back end in the trust store.
    2. Back end service should have the API manager of the trust store.
    - Generate the keys for the back-end.

```
keytool -Keystore back-end.jks -genkey -alias back-end
```

- Export the certificate from the key store

```
keytool -export -Keystore back-end.jks -alias back-end -file back-end.crt
```

- Import the generated back-end certificate to the API Manager trust store

```
keytool -import -file back-end.crt -alias backend -Keystore /valid_jks_store/client-trust store.jks
```

- Export the public certificate from the API Manager's key store.
  keytool -export -Keystore wso2carbon.jks -alias covid2020 -file wso2PubCert.cert

- Import the generated certificate to your back-end truststore
  keytool -import -file puberty.crt -alias wso2carbon -Keystore backend-truststore. .jks

c) API Publisher Security

- API Designer creates either a New API or version of the existing API in the Developer portal. However, it is still not visible to any community, as the API administrator disapproves of that department.
- API administrators (for every sub-organization) review new API definitions, API versions and either approve or deny API by running it against enterprise standards.
- Customize security policy so that every sub organization can leverage and share client credentials instead of user-level credentials.
- Create tenants for every department in an organization. Set new policies for every department in an organization with a specific role call out like publisher, import/export, subscription, etc.
- Define Grant types, Token expiration time, JWT/OAuth token type at either application/user level for every Tenant.
- For any API to be crossly subscribed by other departments, it would involve additional API owners' approval.
- Auto subscription of applications (owner department) to its APIs.
- Create user segmentation, approve API organization, API, versions, and user access to provide specific entitlements.

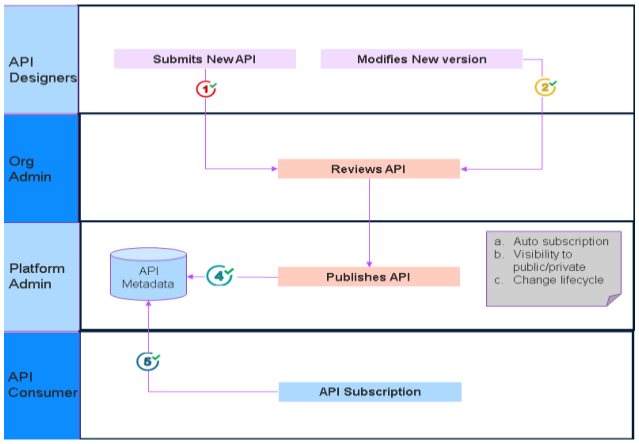


Fig. 5 API Publisher Workflow Strategy

d) Masking and Encryption of CII Data

- Define sensitive data at API definition, at the implementation level (Service Provider), and provide masking requirements wherever possible. This is implemented as a cross-cutting concern (AOP).

- Define custom JSON annotation for sensitive fields in the Swagger/Open API definition. This needs to be defined by the API designer.
  Sensitive fields need to match with the data dictionary of sensitive fields across the organization
  API administrator/owner validates the API against sensitive fields (data dictionary) and approves the same. API gets published to the API repository.
  Masking can be applied in the same format as the real value, e.g. if its social security number xxx-xx-xxxx and date of birth can be XX/XX/XXXX, and regular expressions can be applied at the front gate (API gateway)For Back end services, typically, any microservice can have an Annotation Processor library, which could look at Annotations in implementation and do masking on demand (using JSON Serializer)

- Other sensitive information can be scattered around logs, database end. This can be encrypted and can be decrypted for further processing.

All API manager platform audit logs can be masked or encrypted with the same Serializer. This needs a customized Masking library, which can be a mediation policy/configuration.

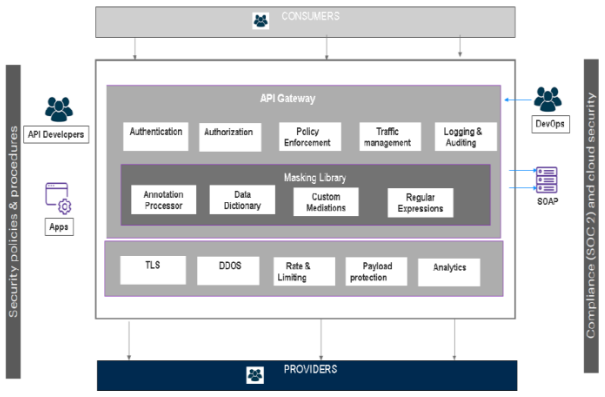


Fig. 6 – Masking of CII data in APIM

e) API Security testing with SAST and DAST

Integrate with CI/CD pipeline, check for SAST, DAST testing during APIs onboard, and back end services.

- Check on back end services dependencies, root-level access, API metadata storage in Persistent Volume, API secrets management with PaaS
- Static Application Security Testing (SAST) tools use a white box testing approach to identify the vulnerabilities. SAST tools are designed to analyze the application's source

code and spot potential issues in the early development stages. This can be leveraged for API – contracts and back-end services.

DAST (Dynamic Application Security Testing) tests it from both API and back end services from a black-box perspective. This tracks the attack vectors on the application and APIs. This can cover SQL Injections, XSS, SSRF out of the box, but we can customize it to provide synthetic attacks (with a valid token) on back end services.

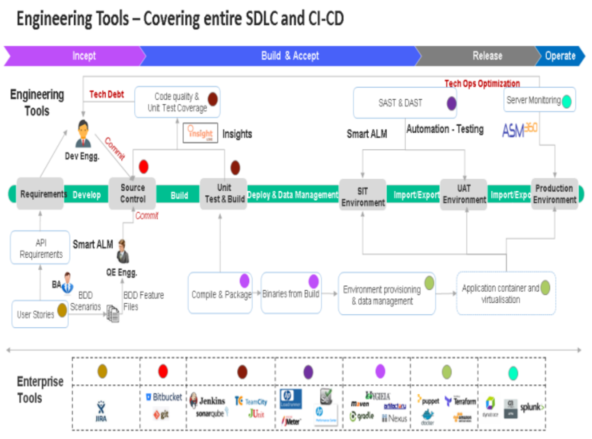


Fig. 7 SAST and DAST with SDLC

f) API Promotion Security Implementation

Enterprises leverage approval workflow within the department or sub-organization to create a new version, API to lower environment. However, enterprises want to leverage CI/CD pipeline (zero human intervention) and promote APIs for a higher environment.

- The CI/CD pipeline is highly secured, with access to only IT teams to initiate the pipeline. Leverage CLI to deploy API to target API server from source (lower) using the import-export feature.
- During the promotion, there is the ability to change the API owner specific to each environment. API owners could be service accounts for each environment.
- Subscription of API's is controlled as in lower environments and carried forward to higher environments.
- For PROD and DR (Disaster Recovery), client credentials can be the same.

The following illustration depicts the proposed security model (highlighted in yellow) with the import /export feature.
All the client credentials are stored in a Secured Vault, provisioning of the APIM platform is integrated with secure fault.

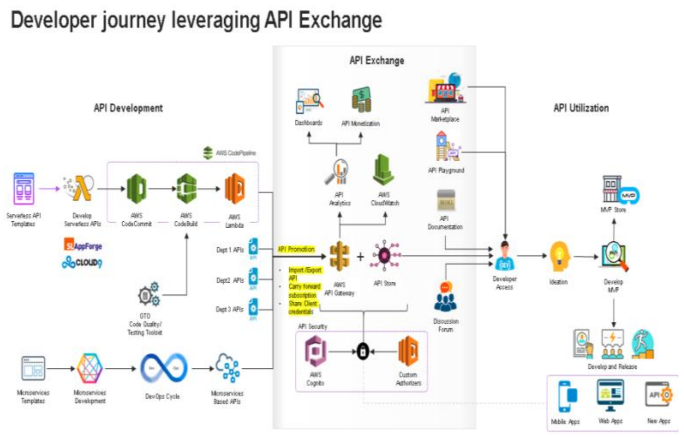


Fig. 8 API Promotion Security

II. CONCLUSION

API has taken centre stage in digital transformation for enterprises now. Being a business enabler has a way of delivering business capabilities to its customers, partners, and internal users.API security is one of the key success factors for large enterprises, as it is a multi-fold implementation (apps, data, network, user identity, roles – user segmentation). A large enterprise's need extends beyond vendor products (in API management) and customizes the same through the API security framework. A well-defined security framework and API management platform (vendor-provided capabilities) can take enterprises long in API adoption and a successful digital transformation journey.

REFERENCES

[1] ProgrammableWeb, http://www.programmableweb.com
[2] Gartner Reports of APIM
[3] SAST, DAST Medium post for reference
[4] https://medium.com/e-t/sast-vs-dast-understanding-the-differences-between-them-406c21d95c79
[5] http://www.internationaljournalssrg.org/ssrg-journals.html
[6] Basic Grant type OAuth reference
[7] https://oauth.net/2/grant-types/
[8] WSO2 promotion for reference
[9] https://wso2.com/library/webinars/automated-api-provisioning-and-promotion-through-cicd/