Module 8-

# Anypoint security

Anypoint Security™ provides advanced defense for your APIs and integrations. Protect sensitive data, stop threats at the edge, and automatically enforce security best practices to protect and govern your application network.

Anypoint Security provides a layered approach to secure your application network. These layers work together to protect both the application network and the network’s individual nodes by controlling access to APIs, enforcing policies, and proxying all inbound or outbound traffic to mitigate external threats and attacks

Anyoint security service is **avaialbe only for RTF** deployment.

Anypoint Security provides two security services

* Edge policy
* Tokenization

1. Edge policy

It is applied at perimeter/ edge pf network

This polices can be added as proxy

The policy that are avaialbe are:

* denial-of-service (DoS),
* IP whitelist,
* HTTP limits, and
* Web Application Firewall (WAF)

1. denial-of-service (DoS)

* prevents attackers from flooding your network to prevent legitimate network traffic to your APIs
* When you create a DoS policy, you configure a time span and action to take when the [error types](https://docs.mulesoft.com/anypoint-security/dos-policy#error_types) you configure are encountered.
* f the policy encounters more errors than your configured threshold coming from the same IP address, the policy can either drop the connection silently or drop the connection immediately and return a 503 error.

1. IP whitelist

* Create list of API that can deploy our application
* Applications that are not in the list are rejected
* Denys all requests by default and allows only requests that are known to be trusted

1. HTTP limits

* The HTTP Limits policy prevents an attacker from sending large messages that consume all your bandwidth
* This policy checks message size and headers if it exceeds the configuration limit it will reject the requests
* Maximum message size, maximum path size, maximum length of single header, maximum length of single trailer, maximum number of header and trailers

1. Web Application Firewall (WAF)

* Web Application Firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet.
* WAF policies are fully integrated with the existing Any point Security policy DoS (Denial of Service). When the WAF policy detects errors, it triggers the thresholds configured in the DoS, which can be optionally configured to take actions such as shaping or blocking traffic for an IP address from a malicious source.
* Request Rule set
  + Scanner detection
  + Protocol enforcement
  + Protocol attack
  + Local file inclusion
  + Remote code execution
  + Remote file inclusion: A complex web application needs to include some files in their current context; attacker can abuse this functionality by including some malicious files in web app or by including server files to view its content. This can lead to sensitive information disclosure, remote command execution.
  + PHP injection
  + Cross-site scripting: is a code injection attack executed on the client-side of a web application. Attacker injects malicious script through the web browser. The malicious script is executed when the victim visits the web page or web server. Steals cookies, session tokens and other sensitive information.
  + SQL injection: SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands
  + Session fixation

1. Tokenization Services

Tokenization is a highly effective way to protect the sensitive data. When you tokenize data, sensitive data elements are substituted with randomly generated non-sensitive data elements. As a result we can protect the sensitive information even if someone is able to breach APIs

Examples of sensitive information that are suitable for tokenization protection include:

* + - Financial Information: Primary Account Number (PAN)
    - Personally Identifiable Information (PII): address, phone number, email
    - Protected Health Information (PHI): medical histories, test result, insurance

It runs inside the network. There are three sub-components in tokenization services. They are:

1.      Masking

2.      Encryption

3.      Tokenization

1.      Masking: Data masking is a technique whereby sensitive data is obscured or obfuscated in some way to render it ‘safe’. It is a one way process, so here we cannot get original value back.

2.      Encryption: It is a format preserving encryption. In this the data is encrypted using some techniques  while the format of the data is maintained. Mule Security offers the

ability to encrypt or decrypt message content in within a Mule flow by utilizing Mule message processors, allowing you to maintain the integrity of your messages

3.      Tokenization: It is the process of masking a value or piece of information that can be considered sensitive data into a token that can be mapped back to its original value using detokenization concepts. The token is displayed in the same format as the original value, but its actual value is not revealed.

Advantages:

* Vaultless operation
* Increased security
* Flexibility
* Performance

**Tokenization Format**

Tokenization formats define how the original format of data coming in is converted to the format you configure when the token is created. For example, you can specify that the token look like the original data source, or whether to tokenize the whole or partial string. Allows to create, update and delete

The data domain configuration options are:

* Preserve first character
* Preserve last character
* Force token characters in illegal range
* Luhn digit test – for credit card number only
* Maximum token length
* Preserve serial number sub-component - or the social security data domain only

1. Secret Manager

* The secrets manager is used to write, read, and manage your secrets, keys, and Transport Layer Security (TLS) certificates within a unique sourc
* It is the central and secure repository to manage the secrets
* **supported on Runtime Fabric and API Manager only**
* uses secure vault technology to store and control access to private keys, passwords, certificates, and other secrets.

**Secrets Group**

A secret group is a unique secret vault to which all your secrets are bound.

Each secrets group has unique encryption keys

Secrets Manager stores your secrets per secret group, per environment, and per business group.

User access and authorization controls are applied at the environment level. A

user who has access on a given environment can access all the secret groups that belong to that environment

two microservices used to implement secret manager

* **Secrets Manager:**
  + This service handles the upload and storing of your secrets.
  + Every time you upload a secret to your vault, the Secrets Manager establishes a reference to it,
* **Secrets Provider:**
  + This is the only service that can read actual secrets
  + . This service is used by the requesting client to consume the secret.

Secret types

1.      **TLS Context**

SSL Security Parameters (ciphers to use, TLS version, and so on).

2.      **Keystore**

Is a server side assets that stores the private keys and certificates with their private and public key

3.      **Truststore**

It is a client side assets that serves as a repository of certification that client should trust

4.      **Certificates**

Public X.509 certificates, which are electronic documents that bind a public key with an identity (hostname, organization, or individual).

5.     **Certificate Pin Set**

A repository of security certificates from other parties that associate a client or host with their expected X.509 certificate or public key.

6.      **CRL Distributor**

An entity that creates and maintains a list of CA certificates that are no longer trusted because their associated private keys, or a signing CA, were compromised.

7.      **Shared Secret Type**

Shared secrets are used for symmetric encryption and decryption, where the secret is known by both the message sender and the message recipient