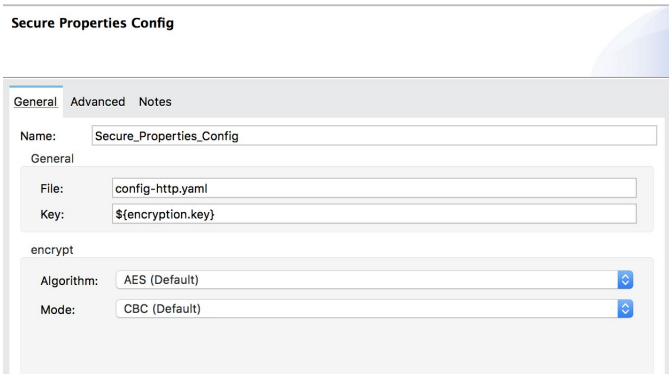
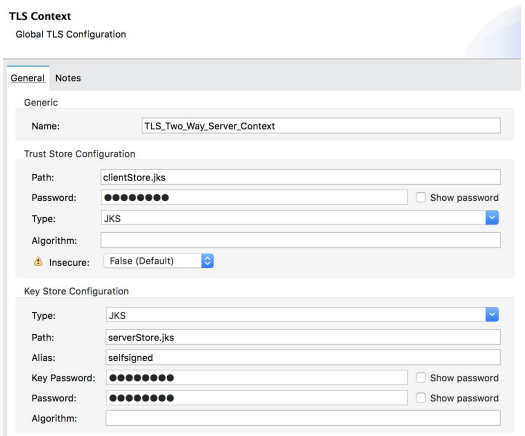
Module 15: Designing Secure Mule Applications and Deployments

Goal





At the end of this module, you should be able to

- Identify Anypoint Platform security concepts and options
- Describe how to secure APIs on Anypoint Platform
- Understand the security needs addressed by Anypoint Platform Edge security
- Differentiate between MuleSoft and customer responsibilities related to Anypoint Platform security
- Evaluate security risks for Mule applications
- Describe how to secure Mule application properties
- Describe how to secure Mule application data in transit

Introducing Anypoint Platform security concepts and options

Anypoint Platform security areas and features

- Securing administrative access to various Anypoint Platform features
- Securing access to deployed Mule applications
  - Securing Mule application connectors
  - Securing the networks to which Mule applications are deployed
- Securing Mule application property placeholder values
- Securing APIs with policies
- Securing data at rest
- Securing data in motion across network edges

Controlling access to Anypoint Platform features

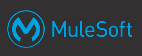
Anypoint Platform access control

- **Roles** can be defined to limit or allow access by various Anypoint Platform users to various Anypoint Platform features
  - Anypoint Platform users can be assigned to roles, and then those users get all the access permissions from those combined roles
  - This allows for distributed administrative groups
  - There are some predefined roles, and other custom roles can also be created
- Every Anypoint Platform username is assigned to exactly one Anypoint Platform **organization**
  - A particular user can be invited to switch to a different organization

An organization owner is the super user of the Anypoint Platform organization

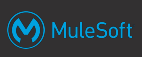
- The **Organization Owner** is the first username to sign up for the organization in Anypoint Platform
  - This is not a role, but a super user identifier for this one user
- The organization owner inherits the **Organization Administrator** role by default
  - The Organization Administrator role has every possible permission
  - Other users can be added to the Organization Administrators role
- Each organization has a **client id** and **client secret** to secure communications with the Anypoint Platform REST APIs

How an identity provider (idP) can be tied in with an Anypoint Platform organization



- By default, Anypoint Platform performs its own identity management
- One external idP can instead be integrated with the Anypoint Platform organization
  - Roles and access control are still enforced inside Anypoint Platform

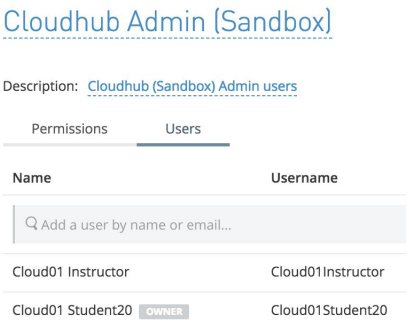
Managing business groups, users, roles and environments



- Organization has business groups, child business groups under business groups, roles, environments and users
- Pre-configured role that business group owners acquire automatically
- Manages business groups
  - Business groups has its client id and client secret
  - Provides isolation of resources
    - But the same user can also be assigned to other roles in different business groups
  - vCores are assigned at business groups makes those vCores available only to the business group and unavailable to the parent organization
  - Business groups has its own environments
  - Deleting business group is NOT recoverable as all resources get deleted

Managing access to business groups

- Access to resources can be controlled by granting appropriate roles in that business group
- To obtain membership to a business group, a user needs to be invited and granted a role
  - When adding users to a role, any user in the organization is eligible to be added
  - Custom roles can be created in a business group

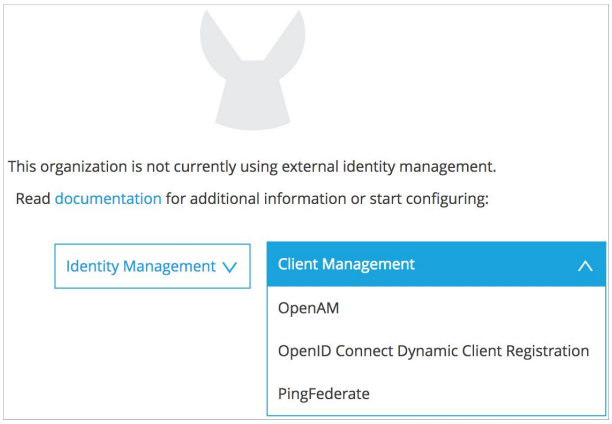


Using an external identity management server with Anypoint Platform

- Can secure Anypoint Platform control plane by configuring
  - OpenID Connect: End-User identity verification by an IdP including SSO
  - SAML 2.0: Web-based authorization including cross-domain SSO
- OpenID Connect supports
  - PingFederate
  - OpenAM
  - Okta
- SAML supports
  - PingFederate
  - OpenAM
  - Okta
  - And many others

Client management options

- Can apply an OAuth 2.0 policy to authenticate client applications
- Anypoint platform supports one IdP for client management per organization
  - PingFederate
  - OpenAM
  - OpenID Connect Dynamic Client Registration (OIDC DCR) compliant identity providers
- Client access for an API is granted for a specific environment and API



Responsibility for Mule runtime security

	MuleSoft-hosted control plane	Customer-hosted control plane
MuleSoft-hosted runtime plane (CloudHub iPaaS)	• MuleSoft secures both planes • Customer secures Mule apps • Customer is responsible for securing communication between CH and on-prem systems	• N/A
Customer-hosted runtime plane	• MuleSoft secures control plane • Customer secures the runtime plane and Mule apps • Shared responsibility for securing communication between on-prem Mule runtimes and control plane • Can configure customer-hosted Mule runtimes to be FIPS 140-2-compliant (does not apply to CloudHub)	• Customer secures both planes • Customer is responsible for securing communication between Mule runtimes and on-prem systems • Can configure the Mule runtime to be FIPS 140-2-compliant

Responsibility for Mule runtime security

- Anypoint Runtime Fabric
  - Is a variant of the customer-hosted runtime plane managed by the MuleSoft-hosted control plane
  - MuleSoft is responsible for securing the Anypoint Platform control plane
  - Customer secures Mule runtime and applications
  - Shared responsibility for securing communication between Runtime Fabric and control plane

Securing Mule application endpoints

How secure communications are typically supported over the internet

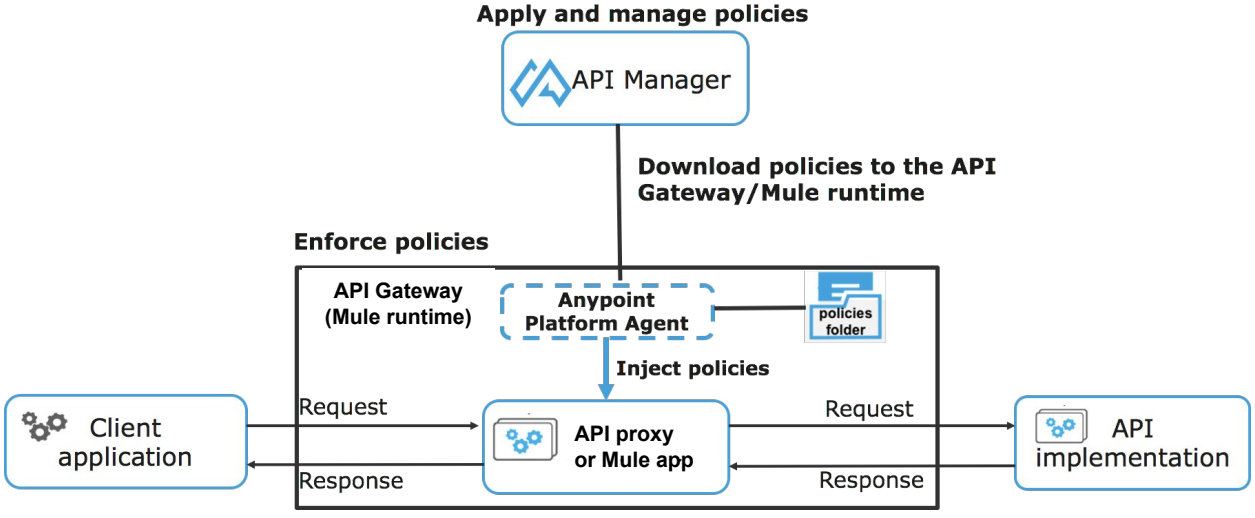
- **Secure communication** is the safe sharing of information that cannot be intercepted by a third party "in the middle"
- Supported using
  - Cryptography
    - Symmetric cryptography
      - Client and server share the same key to encrypt and decrypt
    - Asymmetric cryptography
      - Server issues a public key to the client allowing it to encrypt the message
      - Server keeps a private key which is the only key that can decrypt the message; one key to lock the message and another key to unlock it
  - Digital certificates
    - Uses public/private key certificates signed by trusted authority or self signed for communications

Secure communication protocols supported by MuleSoft

- Mule applications and Anypoint Platform support secure communication using
  - Symmetric and asymmetric cryptography standards
  - Digital certificates
- Supported protocols include
  - HTTPS, TLS, SFTP, FTPS, SMTP/S, IPSec

Securing APIs with policies using API Manager

How policies are enforced using API Manager



API policies apply non-functional requirements to an API

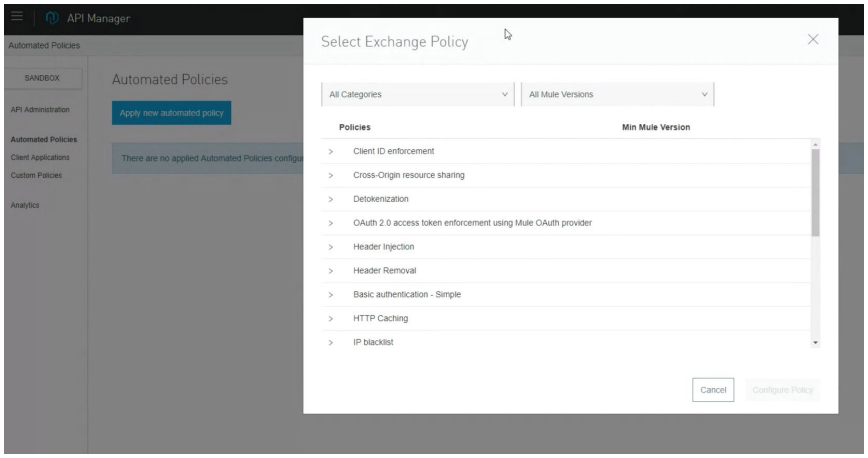
Policy Type	Usage
Client ID enforcement	• Requires authorized client applications to use a client id and client secret
CORS control	• Interacts with API clients for Cross-Origin Resource Sharing • Rejects HTTP requests whose Origin request header does not match configured
Authentication/Authorization	• OAuth 2.0 token enforcement API policies • Basic Authentication: LDAP/Simple • IP-based access control ▪ Blacklisting, whitelisting
Payload threat protection	• Guard against attacks sending over-sized HTTP request bodies
Quality of Service (QoS)	• Rate Limiting: Rejects requests above limit • Spike Control: Queues requests above limit
Caching, logging, and others	• Log parts of the message before or after an API proxy or corresponding backend service is invoked • Caching policies cache the entire backend HTTP response in the API proxy • Header injection and removal policies can be applied to the inbound request or outbound response messages

How policies are enforced using API Manager

- When a policy is configured in the API Manager, the policy is downloaded to the **connected API Gateway or Mule runtimes, inside a /policies folder**
  - Some policies require modifying the RAML API definition. In such cases the API proxy has to be **redeployed** to the API Gateway or Mule runtime
- The policies are then injected into the API proxy or Mule app
  - This way policies can be dynamically changed without re-deploying the API proxy or Mule applications (when the RAML API definition does not change)
  - Does not require restarting the API Gateway or Mule runtimes
- API autodiscovery must be enabled within the application with the API ID
- API client needs permission to access resource protected through API policies

Automated API policies for all MuleSoft-hosted Mule runtimes

- Per API policy, API Manager can configure automatically applying a policy to every CloudHub worker of particular version(s)



Exercise 13-1: View an API's policies

- View an API's policies using API Manager
- View possible policy types and categories available in Anypoint Exchange

Exercise steps

- View an API's policies in API Manager
  - Login to anypoint.mulesoft.com and go to API Manager
  - View American Flights API version v1
  - Look at the existing policies applied to the API

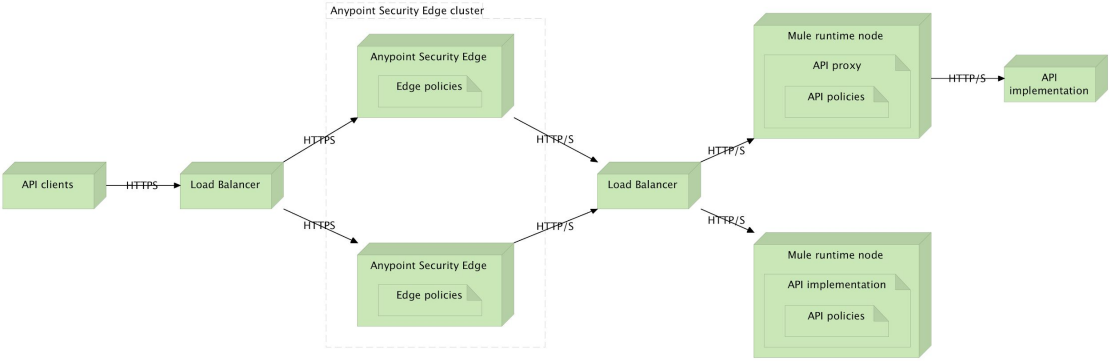
Exercise steps

- View possible policy types and categories available in Anypoint Exchange
  - View all the types of policies available in API Manager from Anypoint Exchange which can be applied to API
  - Identify the types of API categories available in Anypoint Exchange
  - Identify the types of policies available for each API category

Identifying Anypoint Security options

Types of API policies available through Anypoint Platform

- You have already seen how API policies can be defined in API Manager and then injected into Mule applications
  - At runtime, these policies are enforced directly inside a Mule application
- **Anypoint Security** offers other types of policies that can be enforced at the edges of your network for multiple Mule applications

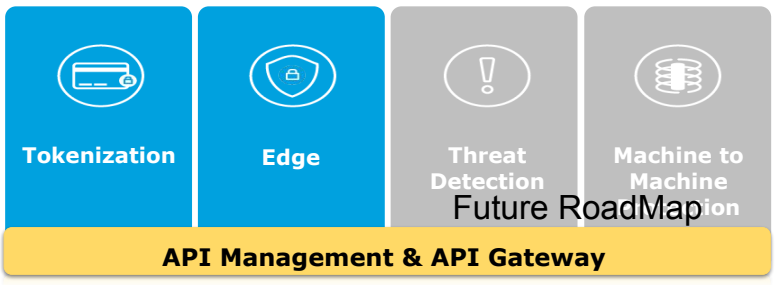


How edge security is used by Mule applications

- Anypoint Security Edge Security is an enterprise solution for **perimeter (edge) level security of APIs**
- Anypoint Security is a **standalone product** deployed outside Mule runtimes
- It is typically deployed in a **DMZ** in a customer-hosted runtime plane
- These edge security policies are applied
  - To inbound requests after they are sent from a calling API client, but before they arrive at the Mule application API endpoints, so **before** API Manager defined policies are applied
  - To outbound responses back to the calling API client, after leaving the Mule application API endpoint, so **after** API Manager defined policies are applied

Anypoint Security

- Adds additional security policies to protect APIs on the edge of your network
- Uses external security servers and services to enforce advanced security related policies
- These API policies are independent of the API policies defined in Anypoint Platform API Manager



Comparing which Anypoint Security options are supported by various runtime planes

	Customer-hosted	CloudHub	Anypoint Runtime Fabric
Edge Security	Yes	No	in 2018
Tokenization and masking	Yes	No	in 2018
Edge Encryption/Decryption	Yes	No	No

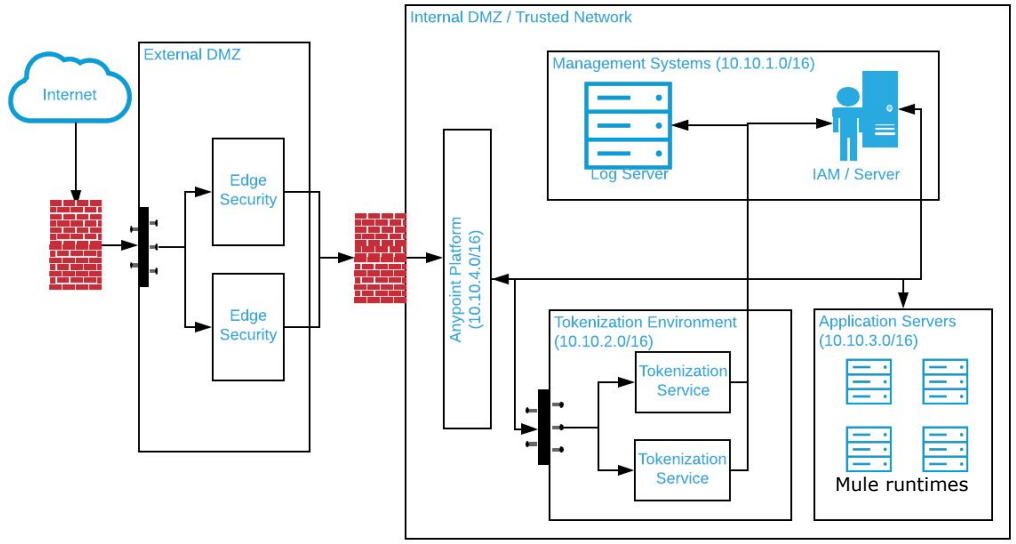
Edge policies vs API policies

	Edge policies	**API policies**
Where policies are applied	Applied in the Edge gateway server	Applied in the API implementation or API proxy app
How the policy implementations works	A single Edge policy can apply to **many API instances**	A single API policy can apply to **exactly one API instance** (deployment)
Available policies	• Service Virtualization • Connection Security and Certificate Management • Content Security • Quality of Service • Application Level(Dos)	• Client ID enforcement CORS control Authentication/Authorization • Payload threat protection • Quality of Service (QoS) https://anypoint.mulesoft.com/exchange/?type=policy

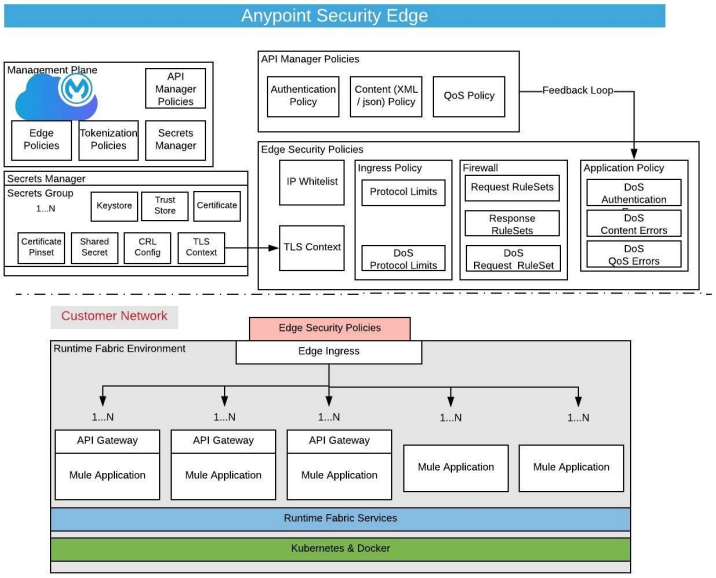
Where Anypoint Security servers are usually deployed

- **Edge Security servers** are usually deployed in a **DMZ** in a customer-hosted environment
  - Provides **edge policies**
- **Tokenization** is usually deployed **inside the firewall** to replace sensitive data with fake data in the same format
  - Such as credit card numbers, social security numbers, or phone numbers

Understanding the interactions between Anypoint Security servers



Anypoint Security Edge



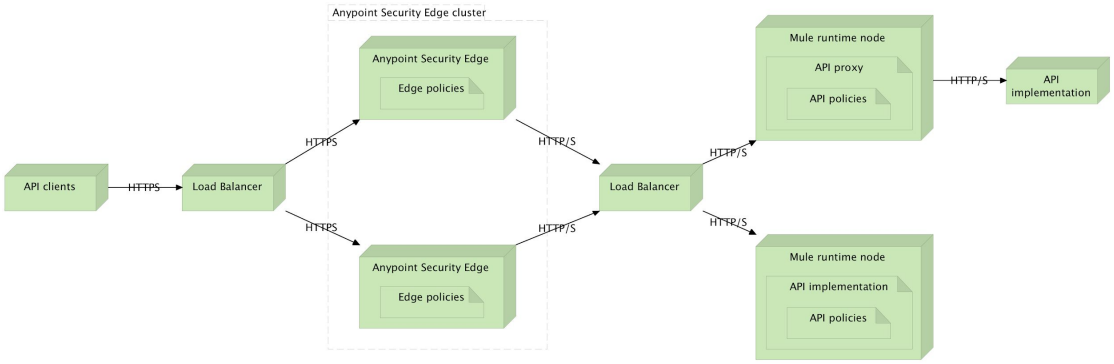
Identifying Anypoint Security edge security features and options

Edge security policies

- A policy is a mechanism for **enforcing filters** on traffic
- These filters generally control things like
  - Compliance
  - Security
  - Quality of service (rate-limiting, throttling)
  - DoS
- Define custom policies by creating **policy definition** and **policy configuration files**
- Apply multiple policies and set the order of their execution

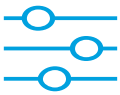
Edge security policies can be changed dynamically at runtime

- Just like API policies defined in API Manager
  - The policies can be dynamically changed without redeploying the API applications
  - Does not require restarting the Mule runtimes



Anypoint security - Edge security overview

MuleSoft


Service Virtualization	Connection Security and Certificate Management	Content Security	Quality of Service	Application Level (DoS)
• Edge Proxy for REST and SOAP	• SSL / TLS Termination • Certificate Management • Chain of Trust Validation / Configuration • CRL Configuration	• Message inspection for Malicious Content, Injection Attacks • Header Validation • Message Size Limits • Message Attachment Size limits	• Resource Consumption Management / Quota & Throttling	• Network Level Denial of Service Protection against too slow or too overwhelming requests • Rate Limiting / Shaping of Traffic (Peak Protection) • Request/Response inspection

Anypoint security - Edge security features

MuleSoft

- Edge Proxy Design policies
  - Abstracts away internal service with a secured endpoint (host/port)
    - Content Security (CAP)
    - Quality of Service (QoS)
    - Denial of Service (DoS)
  - Global Input Server - Support
    - Message Mediation Policy
    - Routing Policy
- Connection & Transport Security
  - SSL/TLS Termination
    - Server Name Indication (SNI)
    - Certificate Pinning
  - Symmetric / Asymmetric Keys configuration
  - Certificate Management
    - Chain of Trust Validation / Configuration
    - CRL configuration
    - Certificate Lifecycle

Anypoint security - Edge security **content attack prevention** (**CAP**)

- Prevents malicious content from reaching a service during runtime execution
- Inspects messages for
  - Pattern Scans
  - SQL injections, XPath injections, and DTDs
  - Limit Message Size
  - Limit # of file attachments / size of each attachment
- Policy will generate a CAP violation if the request matches any of the above criteria

Anypoint security - Edge security **quality of service** (**QoS**) policy

- Manages resource consumption for a user-defined resource, such as a SOAP web service, REST API, or authentication service
- Some sample system resources that are tracked include
  - Request data rate
  - Raw request data rate
  - Response data rate
  - Failed Response rate
  - Message buffer utilization
- Each QoS policy creates a single QoS tracking record for the identity, such as AAA identity, IP address, etc.

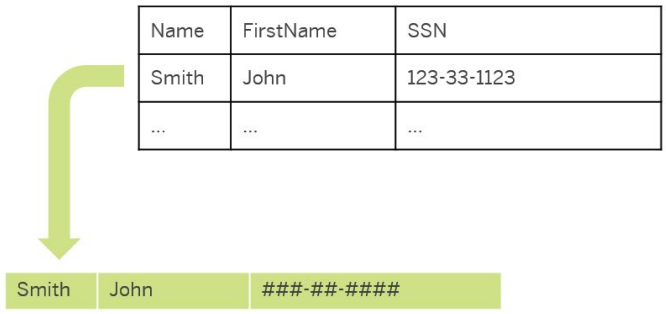
Anypoint security - Edge security **denial of service** (**DoS**) policy

- All policies mentioned earlier provide information to the DoS policy and can be configured centrally for the following actions
  - **Alert** - Sends a message to administrators that a DoS event occurred
  - **Block-Interval** - Rejects messages from a client within an administrator-defined for a period of time
  - **Block-forever** - Permanently block messages from a client unless the administrator removes the restriction
  - **Shape-Interval** - Restricts the rate at which it will accept messages from a client based on a administrator-defined rate.
  - **Shape-forever** - Restricts the rate at which it will accept messages from a client based on a administrator-defined rate unless the administrator removes the restriction

Identifying Anypoint Security encryption and masking features and options

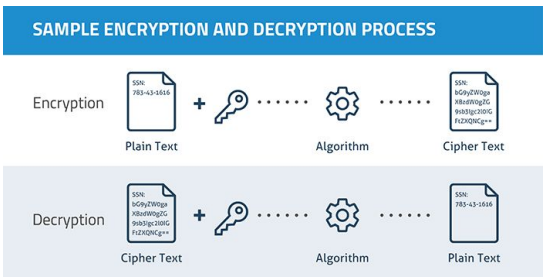
How masking works

- **Masking** protects your data by transforming it into a readable format that's useless to anyone who steals it
- The actual data is replaced by placeholder information
- There is no algorithm to revert the data to its original state
- The real data was replaced and is gone forever



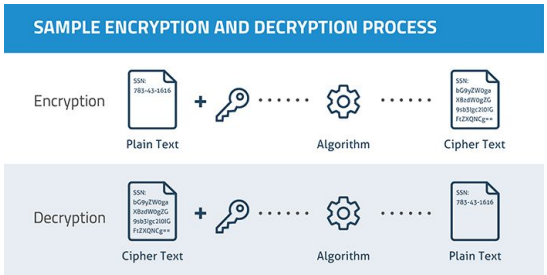
Encrypting data with a symmetric key

- In **symmetric key encryption**, one key is used to both encrypt and decrypt the information
- There can be security related issues
  - If the key is compromised, it can be used to unlock, or decrypt, all of the data it was used to secure
  - How can you securely exchange a symmetric key over a public network?



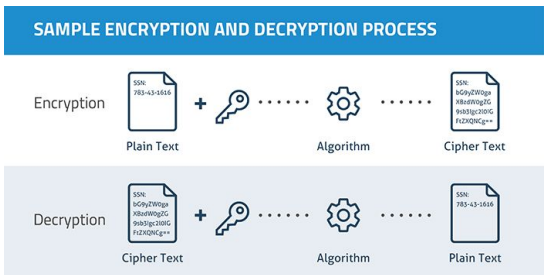
Using asymmetric key (also called public-key) encryption

- **Asymmetric key encryption** was developed to allow multiple parties to exchange encrypted data without needing to manage or exchange the same encryption key
  - The public key can be freely distributed since it is only used to lock the data
  - The private key is used to decrypt the data



In practice, asymmetric encryption is usually only used to safely exchange symmetric keys

- Asymmetric encryption is much slower and more CPU intensive than symmetric key encryption
  - Asymmetric encryption is often used to safely exchange a **symmetric key between two parties** (such as a client and a server)
  - The symmetric key is then used to encrypt and decrypt message payloads
  - A new symmetric key may be periodically exchanged for added security



Identifying Anypoint Security tokenization features and options

Other Anypoint security features

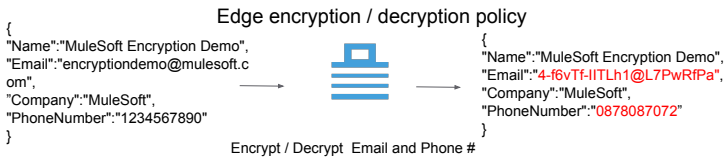
MuleSoft // MeetUp

Tokenization

Application data validation logic work "as is"
No downstream application changes needed
Compliance scope reduction (PCI, HIPAA, GDPR)

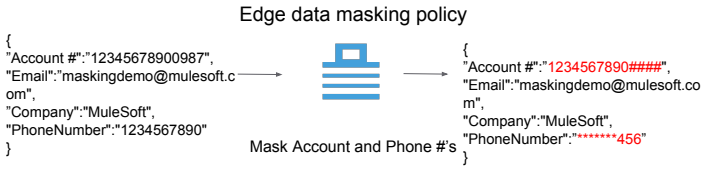
Encryption

Information anonymization
Analytics without exposing sensitive data



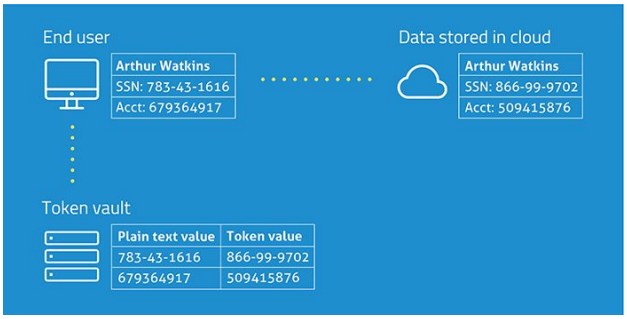
Data Masking

Sensitive data obfuscation
One way process: Cannot get original value back



How tokenization works

- **Tokenization** is the process of turning a meaningful piece of data, such as an account number, into a random string of characters called a token that has no meaningful value if breached
- Tokens serve as references to the original data, but cannot be used to reverse-engineer those values
- There is no key, or algorithm, that can be used to derive the original data from a token

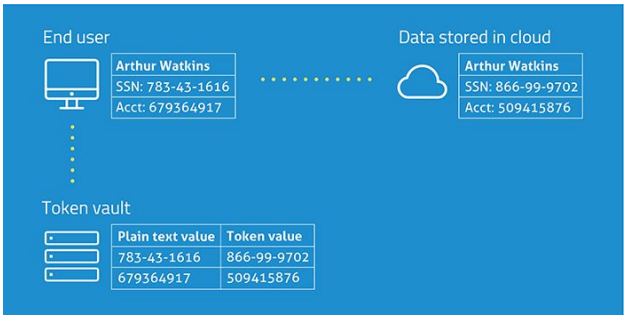


How tokenization works

- Tokenization uses a database, called a **token vault**
  - Stores the relationship between the sensitive value and the token
  - Policy name, operation (tokenization/detokenize), and data is part of the tokenization call
- The real data in the vault is then secured, often via encryption
- Anypoint Exchange has a connector for tokenization

Anypoint Security also implements vaultless tokenization

- Distributed mapping tables shared at each node
- No vault or database is needed to manage or replicate across availability zones / Data centers



Evaluating Mule application security risks

Mule application security risks - Injection

- **Threat** - Consider anyone who can send untrusted data to the Mule application, including users, internal users and administrators
- **Found in**
  - DB processors who accept query text or input parameter such as Execute Script, Execute DDL
    - DB processors accept query text and attackers are likely to send malicious data in query text which can alter data integrity, perform unintended DML or DDL statement on DB
  - XPath extract
    - Xpath extract can execute on unintended element in XML file which can provide sensitive data to attacker
  - Parse template
    - Parse template can provide access to secure properties on server if file system is not protected
  - Headers for HTTP
    - Malicious values in header can cause DoS

Mule application security risks - Injection

- **Prevention**
  - Use a Validation component or other defensive coding
  - Use bind variables in an SQL query instead of string manipulation
  - Avoid dynamic queries
  - Apply Schema validation using APIkit router or XML validate schema components

Mule application security risks - Broken authentication and session management

- **Threat** - Consider anyone who could steal accounts from others
- **Found in**
  - Authentication provider when user authentication credentials are not protected
    - Authentication provider is not storing credentials using hashing or encryption
  - Connectors when password, session ids, credentials are sent on unencrypted connections or embedded in URL
    - Credentials are sent in clear text in URL on non secure transport for connectors
  - Invalidate oauth context
    - OAuth sessions are not invalidated after logout

Mule application security risks - Broken authentication and session management

- **Prevention**
  - Store credentials using hashing or encryption
  - Use secure properties files for customer-hosted deployments
  - Use safely hidden properties for CloudHub deployments
  - Use secure connectors for communication
  - Invalidate sessions promptly
- Note: secure properties and safely-hidden properties are discussed later in this module

Mule application security risks - Security misconfiguration

- **Threat** - Consider anonymous attackers or users who want to compromise systems to gain unauthorized access
- **Found in**
  - ALL connectors
    - Credentials for connectors using default username and password
  - APIkit console
    - APIkit console access is left enabled for API
  - Heart beat or system information or utility apps
    - Heart beat or system information or utility apps are unprotected in application
  - Error handling
    - Error stack trace provide informative information to user

Mule application security risks - Security misconfiguration

- **Found in**
  - Unused message sources, connectors
    - Unused message sources can cause lead unintended usage of application resources
    - Unused ports on HTTP listeners can provide access to application endpoints and resources
- **Prevention**
  - Harden Mule applications and properly lock down the deployed environment
  - Use a strong architecture
  - Periodic security scans and audits
  - Standardized error handling providing error information relevant to the user without providing unneeded or unsecure information about the underlying system

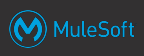
Mule application security risks - Sensitive data exposure

- **Threat** - Consider anyone who can gain access to sensitive data at rest or transit
- **Found in**
  - Application properties
    - Attacker can read clear text files with system credentials which can cause any zero day attack for organization
  - Persistent store such as Object store or file
    - If Object store is not private for sensitive information, then attacker can retrieve sensitive data from Object store
  - Security certificate store
    - If security certificate store is not protected then attacker can have access to unintended services on network

Mule application security risks - Sensitive data exposure

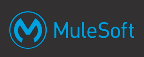
- **Prevention**
  - Use secure properties
  - Use encryption, hashing, or masking for sensitive data (PII or PCI)
  - Protect access to resource using Anypoint Platform enterprise security or OS/VM level grants/permissions

Mule application security risks - Using component with known vulnerabilities



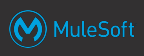
- **Threat** - Some vulnerable components and libraries can be identified and exploited with tools
- **Found in**
  - TLS for HTTP/S, SFTP, SMTP/S and Socket
    - TLS 1.0 is open to **man in middle attack** risking the integrity and authentication of data sent between client and server
  - Custom libraries
    - Untested libraries can exploit application, server, host, for example - Apache Struts vulnerability caused remote attacker to execute arbitrary code on any server running an application built using the Struts framework

Mule application security risks - Using component with known vulnerabilities



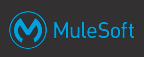
- **Prevention**
  - Use TLS 1.2/1.1 (Do not use deprecated TLS 1.0/SSL 3.0)
  - Use only approved enterprise libraries
  - Perform static and dynamic code analysis of Java sources or byte code

Exercise 15-2: Identify security threats exposed by a sample application



- Review and identify security threats exposed in a Mule application

Exercise steps



- Import sample Mule application (exercise-15-2.jar) in your student files
- Go to "security-misconfigurationFlow"
- Verify various components and property files in flow
- Check for below security threats
  - Security misconfiguration
  - Sensitive data exposure
  - Broken authentication and session management
- Identify threats exposed by flow in each category

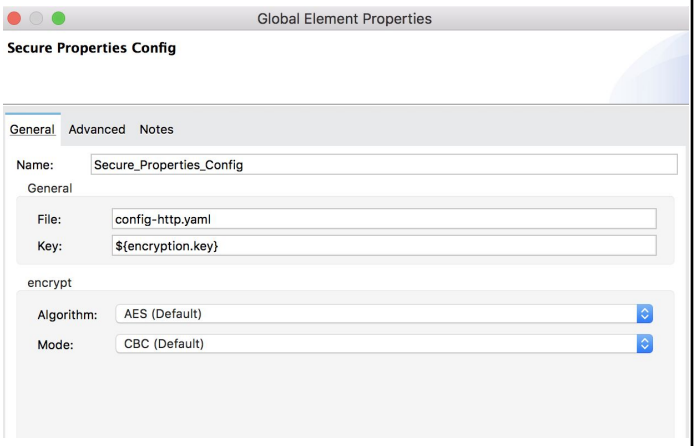
Exercise steps

- Security misconfiguration
  - Password, security key are in clear text
  - Unused/vulnerable properties in property file
  - Misconfiguration of properties
- Sensitive data exposure
  - Data stored in database in clear text
  - Data transmitted in clear text
- Broken authentication and session management
  - User authentication credentials are not protected
  - Credential can be guessed
  - Session did not timeout
  - Credential sent over unencrypted communication

Securing application properties

Securing application properties

- Secure Properties Config encrypt properties and store the encrypted text in secure properties files
- Supports multiple encryption algorithms and mode
- Encryption key for encryption sets as system property or env variable
- Define one secure config for each secure property file
- Install extension as dependency in pom
- AES algorithm with CBC mode is default



Define secure config

- Each secure property file needs one secure config

```
<secure-properties:config name="Secure_Properties_Config"
file="config.yaml" key="${encryption.key}" >
        <secure-properties:encrypt algorithm="AES" mode="CBC"/>
</secure-properties:config>
```

Securing property file

- The **Secure Configuration Properties** module supports both YAML configuration files and Java properties files
- Encrypted value is defined in **![**value**]** in quotes

```
db:
  host: "localhost"
  port: "3306"
  user: "root"
  url: "training"
  password: "![DJqeCF2q+gwuHPxNw6+apA==]"
```


Accessing secure property

- The **secure:: prefix** is use to access encrypted and clear-text values from secure property files

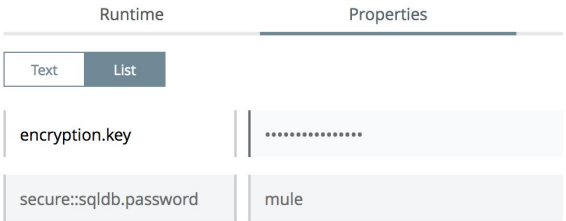
```
<db:config>
        <db:my-sql-connection host="${secure::db.host}"
        port="${secure::db.port}" user="${secure::db.user}"
        password="${secure::db.password}"
        database="${secure::db.url}" />
</db:config>
```

How to update secure properties

- Can be updated at design or deployment time
- To override or update properties using the Runtime Manager console's property tab (or REST API), values are entered as plain text, NOT as an encrypted value
  - Runtime Manager cannot access the decryption key
  - CloudHub deployments can safely hide the new value

Safely hiding application properties in CloudHub Mule application deployments

- CloudHub supports **safely hiding** Mule application properties
- Properties are encrypted and stored in the online CloudHub properties service (database)
  - The value can then never be seen in the Runtime Manager GUI



mule-artifact.json

```
{
  "configs": [
    "scalability-reliability-module.xml"
  ],
  "secureProperties": ["secure::encryption.key"],
  "redeploymentEnabled": true,
  "name": "scalability-reliability-module",
  "minMuleVersion": "4.1.2",
  "requiredProduct": "MULE_EE",
  "classLoaderModelLoaderDescriptor": {
    "id": "mule",
    "attributes": {
      "exportedResources": []
    }
  },
  "bundleDescriptorLoader": {
    "id": "mule",
    "attributes": {}
  }
}
```

Setting key for secure property config

- On-premises
  - Set from JDK system properties
    - `Mule -M-Dencryption.key=Mule`
  - Set from environment variables from OS
    - varies with OS
  - Set from wrapper.conf in <Mule_HOME>/conf
    - wrapper.java.additional.<n>=-Dencryption.key=Mule
- CloudHub
  - Set from Runtime Manager console's property tab
  - Best practice is to set key from Runtime Manager console and hide as application property by listing under secureProperties key as comma separated list in mule-artifact.json

Exercise 15-3: Prevent misconfiguration of secure properties for a Mule application

- Prevent the security misconfiguration identified in the sample Mule application exercise-15-2.jar in the solution of Exercise 15-2

Exercise steps

- In CloudHub, import exercise-15-3.jar for analysis
- Verify the following conditions:
  - Password, security key are in clear text
  - Unused/vulnerable properties in property file
  - Misconfiguration of properties

Exercise steps: How to prevent password, security key in clear text

- Choose Configuration Property or Secure Configuration Property
- Decide algorithm and mode
- Choose security key
- Verify the security key is plain text and readable
- Discuss ways to secure the security key

Exercise steps: How to prevent unused/vulnerable and misconfiguration properties

- Review the Mule application for unused/vulnerable properties in the property file
- Review the Mule application for misconfiguration of properties

Exercise solution: Define Security config

```
<secure-properties:config name="Secure_Properties_Config"
        file="config-http.yaml" key="${encryption.key}" >
   <secure-properties:encrypt algorithm="AES" mode="CBC"/>
</secure-properties:config>
```

Exercise solution: How to secure key

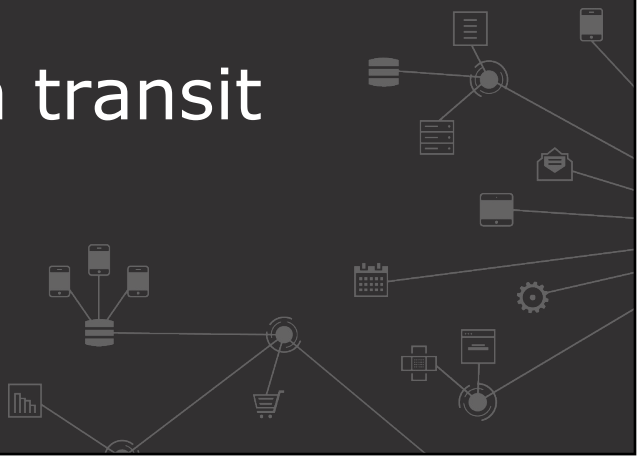
wrapper

- Analyse options to secure the key
  - Set property placeholder values through a JDK system property from the command line
  - Set from the wrapper.conf file
  - Set from an env variable in the operating system
  - Set from the Runtime Manager console's Property tab
- Choose an option to secure the key
  - CloudHub does not provide control over JDK system property, wrapper.conf or OS env variable and best option to secure key is set from Runtime Manager console's property tab

Exercise solution: Prevent security misconfiguration for sample application

```
db:
  host: "localhost"
  port: "3306"
  user: "root"
  url: "training"
  password: "![DJqeCF2q+gwuHPxNw6+apA==]"
```

Securing data in transit



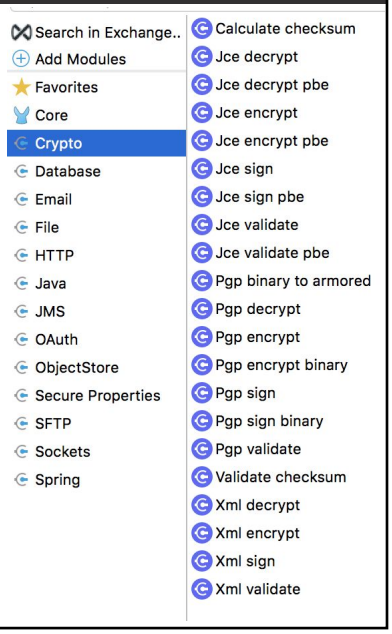
How to secure data in transit

MuleSoft

- Network related security responsibilities usually carried out by ops
  - Configuring network protocols to protect data in transit
    - HTTPS, SSL, TLS
  - Configuring Anypoint Edge security can also protect data in transit
  - These options are covered in the next module
- Security responsibilities usually carried out by Mule developers
  - Needing to encrypt data within the Mule application
    - Mule provides a Crypto security module to encrypt and decrypt data inside a Mule application

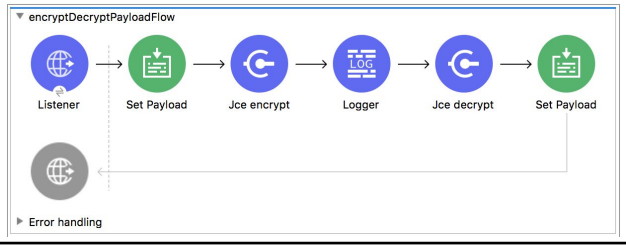
How developers secure data in Mule applications

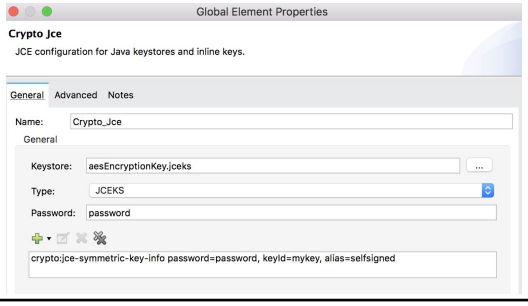
- The Mule Cryptography module supports a Java Cryptographic architecture for
  - Symmetric or asymmetric encrypting/decrypting
  - Signing and signature validation of a payload or part of a payload
- Supports
  - Java cryptography extension (JCE)
  - Pretty good privacy (PGP)
  - XML
- DataWeave also has a crypto module for hashing payload
  - `import dw::Crypto` in DW to use Crypto functions



How Crypto works

- For encrypting/decrypting and signing and signature validation of a payload
- To generate keystore file
- Java Keytool generate keystore file using
  - `keytool -genseckey -alias selfsign -keyalg AES -keystore aesEncryptionKey.jceks -keysize 128 -storeType JCEKS`
- Mule Crypto processors use JCE configuration with generated key store file





Summarizing security best practices

Security best practices

MuleSoft

- Use "Least privilege access" for Anypoint platform
- Do not rely on default security settings for platform or connectors
- Default error handling should be avoided as it provides lot if information to attackers
- Audit/logging should mask PII data
- Logging setting for application, runtime carefully set for environment

Summary

Summary

MuleSoft

- Anypoint platform security provides various security features
  - Identity management
  - Client management
  - Mule secure property
  - Hiding application property
  - Mule Crypto module
  - DataWeave Crypto module
  - TLS for connectors
  - Anypoint Edge

Summary

- The customer needs to specify the level and type of security required by various integration solutions
- Anypoint Platform enables customers to build highly secure Mule applications that leverage enhanced inbuilt security in the platform
- The Mule runtime is built on top of standard Java security
- A Mule runtime can be configured to comply with the FIPS 140-2 standard

References

- Java SE Security
  - https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html
- Java Cryptography Architecture Standard Algorithm Name Documentation for JDK 8:
  - https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html
- Java Cryptography Architecture (JCA) Reference:
  - https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html
- SSL for Apache Active MQ
  - https://support.Mulesoft.com/s/article/How-to-enable-SSL-in-Apache-Active-MQ