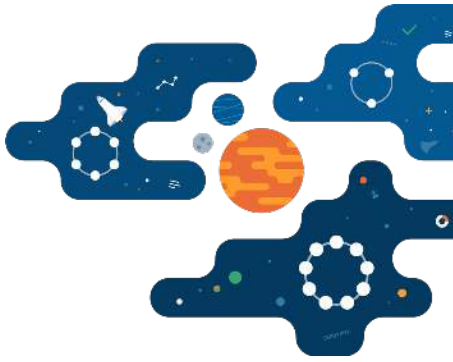
WHITEPAPER



Security Best Practices for Apache Pulsar

Apache Pulsar is rapidly becoming the go-to technology for enterprises that want to modernize their event driven architectures to provide real time data processing capabilities across their organization.

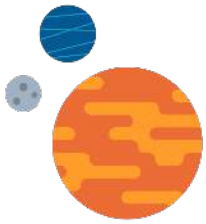
As the volume of real time messages and events grows, so does the sensitivity of the data being processed by Apache Pulsar. In turn, the need to ensure that Pulsar is configured securely and that the necessary safeguards have been implemented to prevent data breaches and other risks to this data is a critical factor for enterprises when adopting this technology.

Fortunately, Apache Pulsar has been built from the ground up to provide a foundation of security. DataStax has extended this even further to provide an enterprise ready distribution of Apache Pulsar through the DataStax Luna Streaming product.

DataStax

01

A Layered Approach to Security



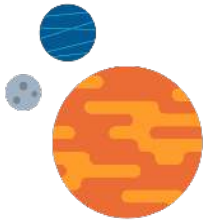
In this white paper we will walk through common aspects of security that you should take into account when implementing Apache Pulsar along with guidance for how to configure your Pulsar instance to ensure secure operations:

- **Infrastructure:** With an emphasis on Kubernetes (k8s) based deployment, what considerations and techniques should you employ to secure your Apache Pulsar deployment?
- **Network Security:** What can you do to ensure all communication within Pulsar occurs over secure channels?
- **Data Security:** How should you secure message data and what steps would you need to take to protect messages based on the sensitivity and classification of the data they contain?
- **Identity & Access Management:** How can you configure Apache Pulsar to work with your organization's single sign on (SSO), identity provider, access control manager to provide authentication and authorization?

This whitepaper will also provide you with a guide to securely configure and operate DataStax Luna Streaming using many of the standard features found in Apache Pulsar as well as capabilities such as SSO integration and enterprise authentication/authorization which are exclusive to DataStax Luna Streaming.

02

Infrastructure Security on Kubernetes



Kubernetes provides general guidance around *best practices for securing a k8s cluster.*

Comprehensive Kubernetes security is outside the scope of this document, however there are a number of Pulsar-specific configurations that you should be aware of when running your cluster on Kubernetes. Luckily, the ***DataStax Helm Chart installer for Luna Streaming*** gives you a seamless way to quickly stand up your Apache Pulsar cluster using a configuration that automatically applies these configurations.

Container Privilege Levels

Like any enterprise software, you should strive to implement a policy of least privileges to ensure that the user the software is running as has the necessary permissions to carry out its tasks, but no more.

When using DataStax Luna Streaming, the default configuration will ensure that your Pulsar containers run as the pulsar user rather than root. This user is configured with group permissions to access the necessary locations on the container's filesystem and to invoke the necessary commands, but does not have superuser access on the container.

Audit Logging

Apache Pulsar using log4J as its logging framework. By default, Pulsar will log failed authentication events from the following classes/log levels:

```
org.apache.pulsar.broker.web.AuthenticationFilter / WARN
org.apache.pulsar.broker.service.ServerCnx / INFO
```

Likewise, authorization failures are captured by the following class / log level:

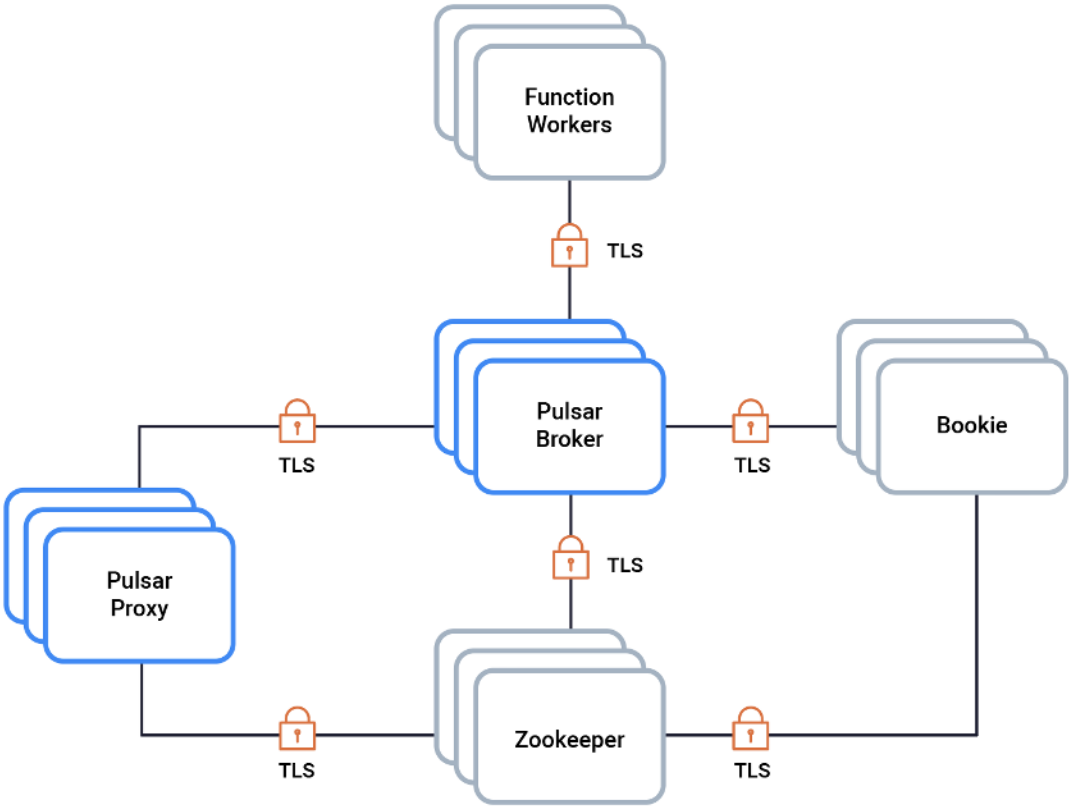
```
org.apache.pulsar.broker.service.ServerCnx / WARN
```

As part of your ongoing monitoring of Pulsar, you can use a third party product to monitor the content of these logs to surface potential security issues such as brute force attacks which you can then appropriately respond to.

03

Network Security

Securing Connections Between Pulsar Components



If the distribution of Apache Pulsar you are using is DataStax Luna Streaming, then you will find that configuring TLS between components is a matter of simple configuration in the ***values.yaml*** file you supply as part of your Helm chart installation.

In this file you will find a TLS section which allows you to enable TLS selectively. The relevant section is shown here:

```
enableTls: false
tlsSecretName: pulsar-tls

tls:
  zookeeper:
    enabled: false
  # Enable TLS between broker and BookKeeper
  bookkeeper:
    enabled: false
  # Enable TLS between function worker and broker
  function:
    enabled: false
  # Enable TLS between WebSocket proxy and broker
  websocket:
    enabled: false
```

Helm chart values to configure TLS between Apache Pulsar components.

Securing Apache Pulsar Admin API Access

Pulsar provides a REST API endpoint which is used by the pulsar-admin command line interface as well as the Java administrative API. By default, the HTTP endpoint exposed by Pulsar is unencrypted. Rectifying this is simply a matter of configuring the appropriate TLS certificates and configuring them within Pulsar.

To accomplish this, you will need to configure the webServicePortTls property in the ***broker.conf*** file to enable TLS.

One you have configured TLS, you will also need to enable authentication on this endpoint by setting the following properties in broker.conf:

```
authenticationEnabled=true
authorizationEnabled=true
authenticationProviders=<desired provider>
```

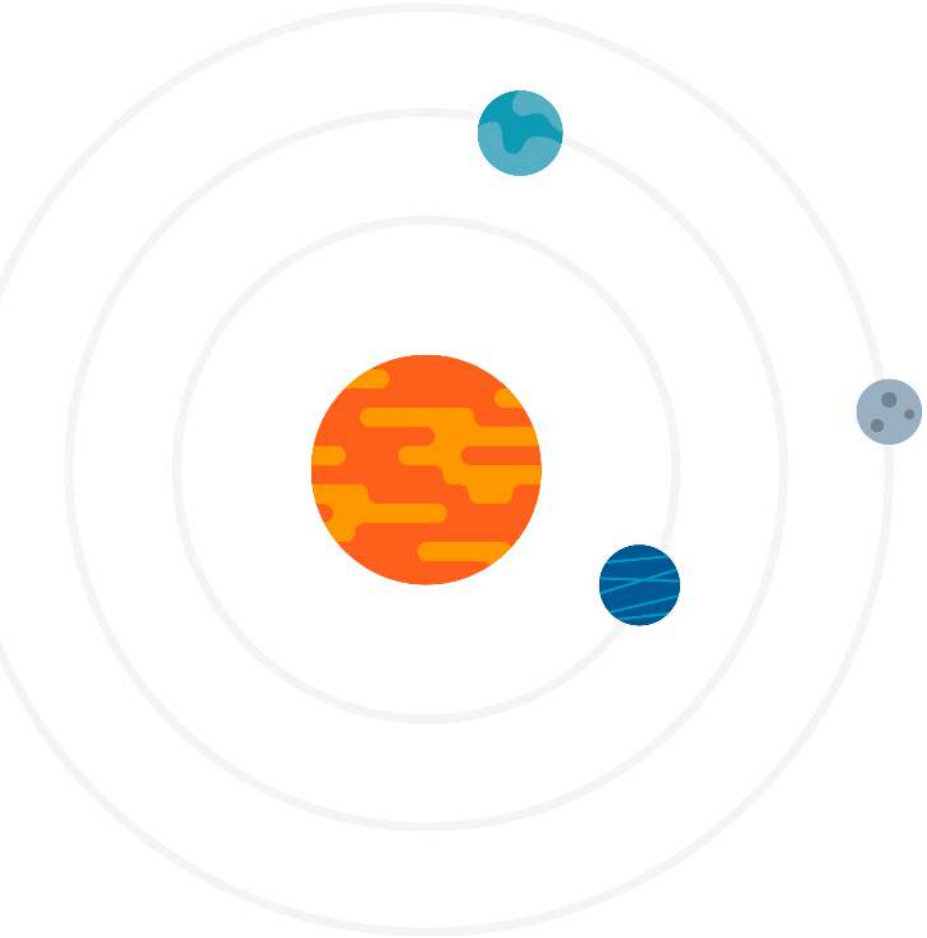
An example of configuring secure access on the broker can be found in the ***Pulsar documentation.***

Configuring a Secure Channel for Clients

Pulsar clients communicate with the Pulsar brokers using the Pulsar binary protocol which is based on Protobuf. Clients establish a connection either directly to the broker(s) or to the Pulsar proxy which then routes the client's request to the appropriate broker to handle the request.

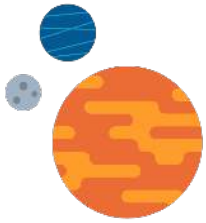
By default, Pulsar configures brokers to communicate over unencrypted connections. To change this, you must configure the ***broker.conf*** to enable TLS. This is done by configuring the brokerServicePortTls property to have a value of true. Additionally, you can configure the appropriate settings in that same file to instruct Pulsar to use the TLS certificates which have been issued by your organization's certificate authority.

Likewise, when using the Pulsar proxy component to serve as router to brokers, a similar configuration can be made in the ***proxy.conf*** file to ensure that clients that connect to Pulsar do so over a secure channel.



04

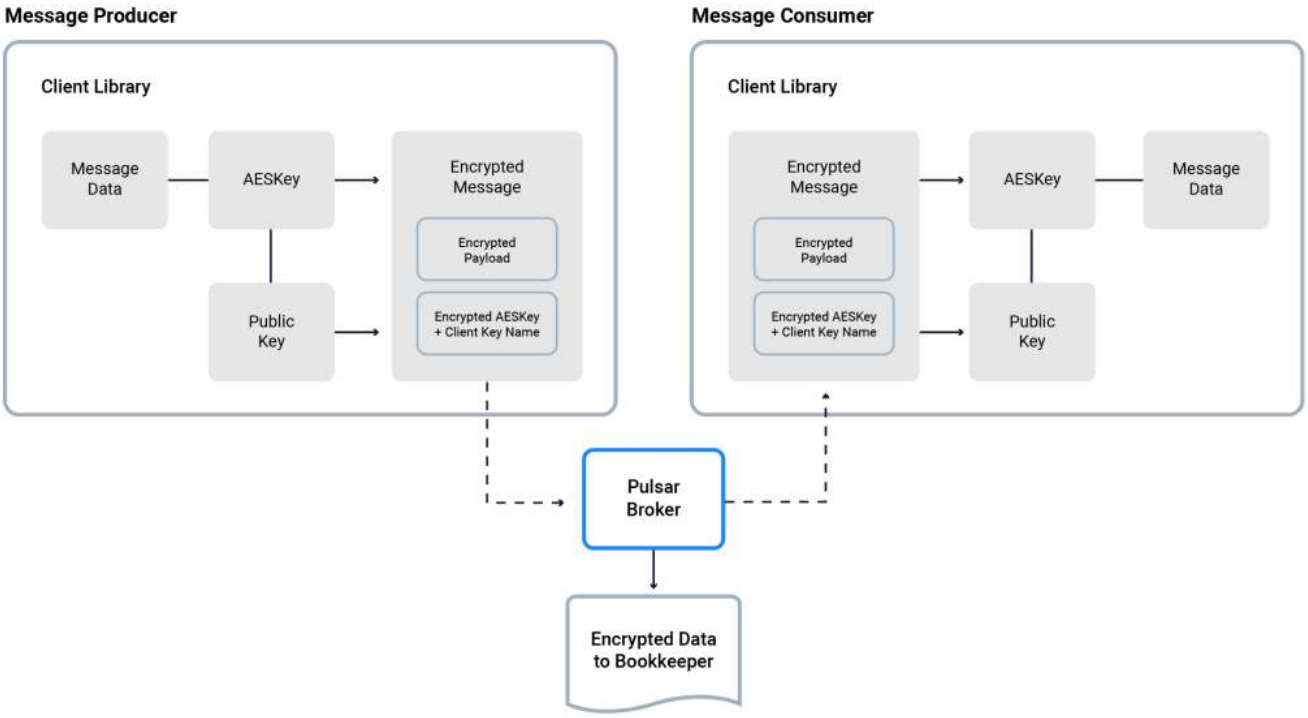
Ensuring Data Security and Confidentiality



Encrypting Message Data

Currently BookKeeper does not provide support for encrypted ledger data. This means that there are two options for encrypting message data at rest. The first is to rely on encryption mechanisms at the physical storage layer such as disk level encryption. These solutions will vary from provider to provider, however many common solutions encrypt disks only when they are unmounted, providing limited protection against an attacker who has access to the VM/container where the disks are mounted.

Luckily, Pulsar supports end-to-end encryption of message data. Using this approach, the message producer encrypts messages before publishing them to Pulsar. The encrypted ciphertext is stored as a byte array in BookKeeper and decrypted by message consumers.



This approach offers an airtight approach to encrypt message data with the tradeoff of complexity for managing consumers and for accessing and replaying historical stream data; by default Puslar will rotate the encryption keys every 4 hours.

Message Retention and Purging

One standard way to prevent data leaks is to simply delete data that is no longer required. Pulsar provides several mechanisms to assist with this. The first is that for a given topic, you have the option of configuring it as either persistent or non-persistent (ephemeral). Topics of either type can be easily identified by inspecting the fully qualified topic name. Persistent and non-persistent topics will begin with `persistent://` and `non-persistent://` respectively.

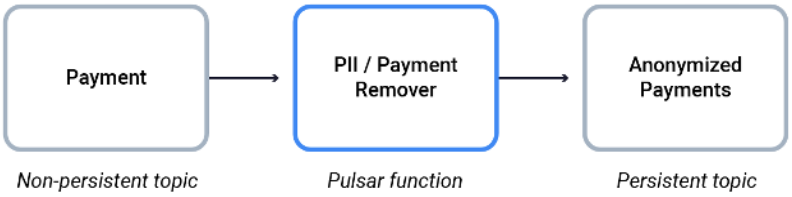
Non-persistent topics will never have their data persisted to BookKeeper. This can be convenient for use cases which need to be optimized for performance or for cases where message data loses its value if not immediately processed. This same capability can also be beneficial for cases where you want to avoid persisting sensitive data all together such as in the context of payment use cases or where patient medical data is involved.

For cases where message persistence is desired, but where you want to purge data after a certain period of time, you can configure message retention policies at the namespace level within Pulsar. This is convenient if you have data which is subject to regulatory compliance such as GDPR where the commonly recommended practice is to retain applicable personal data for only as long as necessary before purging it from your system. Using this capability in Pulsar you can set a retention period that is appropriate for your organization, say 60 or 90 days, after which message data will be automatically purged from the system.

Data Masking Using Pulsar Functions

Apache Pulsar includes a capability known as Pulsar Functions for in-stream processing of message data. This feature provides a simple mechanism which can be used to preserve the benefits of historical stream data while adhering to best practices around data privacy and retention. For example, from a data science perspective, it may be beneficial to retain a historical record of order and payment messages from an ecommerce system. However, these messages may contain PII or payment card data which introduce risks if stored indefinitely.

Pulsar functions allows you to programmatically modify message data and pass it along to a new topic:



Using this technique you can implement data masking, data scrubbing and other approaches to adhere to your organization's data handling procedures, while enabling deeper analysis of event stream data from a historical context.

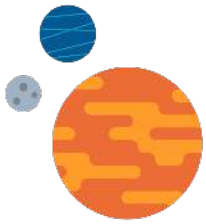
Message Validation Using Schemas

Ensuring message data is valid provides benefits from a data cleansing perspective, but also offers security benefits as well. While we typically think of things like SQL injection attacks as a concern for web applications, a similar attack surface exists anywhere that data is taken as an input and turned into a database command to modify the state of a data store. Likewise, an attacker could try to create an overflow condition by sending in very, very large amounts of data in a message to overwhelm your consumers and trigger downstream problems throughout your organization.

Pulsar provides support for Avro and native Protobuf based schemas. Using these allows you to ensure type safety of message payloads along with finer grained control over individual data elements. For example, using Avro it is possible to validate JSON payload data using regular expressions, value ranges and other common approaches to ensure data adheres to the format and acceptable limitations you set.

05

Identify and Access Management



Authentication in Pulsar

Pulsar uses a pluggable authentication model that includes support for the following authentication mechanisms:

- **TLS Authentication** – use client certificates to authenticate clients.
- **Athenz** – leverage Athenz tokens for client authentication
- **Kerberos** – SASL based approach for authentication using Kerberos
- **JSON Web Token Authentication** – JWT signature based authentication tied to subject (sub) claim
- **OAuth2** – use the OAuth2 client credentials flow to obtain tokens for authentication.

Each of these approaches is backed by an AuthenticationProvider implementation which is responsible for matching the subject in Pulsar to the role which that user has been granted.

DataStax Luna Streaming extends these options with more full featured enterprise authentication capabilities. Luna Streaming ships with Keycloak, a full featured, open source identity and access management solution. Keycloak allows you to integrate a wide range of common authentication solutions into Pulsar including LDAP, SAML, OpenID Connect and others. Keycloak also supports user federation using Kerberos and LDAP. This ensures that developers can use standard technologies already in use within your organization to address common security requirements.

Authorization in Pulsar

Pulsar manages permissions at a namespace level and uses the concept of a role to specify what actions the role is allowed to perform within the namespace. The allowed actions are either `produce` or `consume`.

Additionally, Pulsar has a predefined set of permissions called `superuser` at the cluster level and a predefined set of permissions called `admin` at the tenant level.

Superusers, as the name suggests, are allowed to perform any action within the cluster including administering tenants, namespaces and other resources as well as producing to/consuming from any topics anywhere in the cluster.

To specify the roles which should be given superuser permissions, you must supply them as part of your `broker.conf` file. If you are using the DataStax Luna Streaming helm chart, you can also specify this in the ***values.yaml*** file:

```
superUserRoles=my-super-user-1,my-super-user-2,my-proxy-role
```

Admins are limited to the tenant and have the ability to administer namespaces and other resources within the tenant as well as producing to/consuming from any topics anywhere in the tenant. To grant a role admin permissions on a tenant, you can specify this at tenant creation time:

```
bin/pulsar-admin tenants create my-tenant \
  --admin-roles my-admin-role \
  --allowed-clusters us-west,us-east
```

If you want to change the roles which have been granted admin permissions on a given tenant you can do that as well using this command:

```
bin/pulsar-admin tenants update my-tenant \
  --admin-roles original-admin-role, new-admin-role
```

For finer grained access at a namespace level, the following command will allow you to specify which roles are allowed to consume from/produce to topics in the namespace:

```
bin/pulsar-admin namespaces grant-permission my-namespace \
  --actions consume \
  --roles consumer-role
```

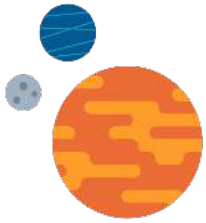
The allowed actions are either `produce` or `consume`. These can also be set when creating a namespace.

Monitoring Authentication Metrics

Pulsar exposes metrics that give you visibility into ***authentication events.*** These metrics are exposed in the standard Prometheus format, making it straightforward to connect Pulsar with existing monitoring solutions in use within your organization. In addition to the audit logging suggestions outlined above, monitoring these metrics can give you visibility to incidents where authentication failures may spike and where further investigation is likely warranted.

06

Lunar Streaming Security Differentiators



Security is a key part of implementing and adopting any technology. While Apache Pulsar provides a strong foundation for security, these capabilities alone may not be sufficient to meet the needs of enterprise security standards in place at most large organizations.

While we've covered many aspects of securing your Pulsar instances in this paper, there are other out of the box capabilities which are exclusive to DataStax Luna Streaming as well:

- **Vulnerability Scanning** – Every release of DataStax Luna Streaming undergoes vulnerability scans to ensure there are no known severe vulnerabilities in the release.
- **Rapid Patches** – When security vulnerabilities are detected DataStax is able to release patches outside the normal OSS release process which can result in faster time to remediation for security issues.
- **Enterprise Authentication / Authorization** – DataStax Luna Streaming ships with support for enterprise SSO and access control standards such as LDAP, SAML, OpenID Connect and others.

Contact / Feedback / Questions

If you have feedback or questions about the content of this whitepaper or Pulsar in general, we'd love to hear from you! You can reach the DataStax team who is focused on Apache Pulsar by emailing ***pulsar-team@datastax.com.***