appviewx

Post-Quantum Readiness Guides

Version: 2025.0.0.0

Copyright AppViewX, Inc.

Trademarks

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface 6

Revision History 6

About this Guide 6

Audience 6

Third-Party Software Acknowledgments 6

Text Conventions 6

Chapter 1. Understanding PQC and the Need for It 7

AppViewX's Quantum Trust Hub Platform 7

Chapter 2. Prerequisites for Working with the AppViewX Config Scan and Code Scan Agents 9

Chapter 3. Supported Ecosystems for PQC Assessment in AppViewX 12

Unified Scan 12

Code Scan 14

Chapter 4. Configuring and Executing the AppViewX Config Scan Agent 16

Linux-based Installation of the Config Scan Agent 17

Docker-based Installation of the Config Scan Agent 21

Configuration File Prompts 27

Additional Commands for Config Scan Agent Configuration 30

Chapter 5. Configuring and Executing the AppViewX Code Scan Agent 34

Deployment Types 34

AppViewX Code Scan Agent: Execution Flow 34

Creating the Configuration File for Agent Set Up 35

Additional Commands for Code Scan Agent Configuration 40

Configuration File Prompts 42

Integrating the AppViewX Code Scan Agent with CI/CD Pipelines 45

Executing the Code Scan Agent 58

Sample YAML files for CI/CD Integration 62

Sample Linux Executable YAML File 63

Sample Docker Executable YAML File..... 63
Chapter 6. AppViewX Quantum Trust Hub User Guide for PQC Readiness..... 65
Key Features of the Quantum Trust Hub..... 65
Licensing and Access Control for PQC in AppViewX..... 67
Viewing the Quantum Trust Hub..... 71
Understanding the Quantum Trust Hub Dashboards..... 71
Organization Overview..... 72
List of Scans..... 82
Code Scan..... 83
Configuration Scan..... 84
Certificate Scan..... 84
Quantum Trust Hub Inventory..... 84
Quantum Trust Hub Policy..... 85
Chapter 7. Integrating and Operating the PQC Assessment Tool..... 86
Chapter 8. Scanning Cryptographic Assets for PQC Readiness..... 87
Quantum Trust Hub: Code Scan..... 87
Quantum Readiness Score..... 88
Code Scan Count Cards..... 89
Quantum Readiness Posture..... 90
Cryptographic Library Sources..... 90
Quantum Readiness by Crypto Library..... 91
Quantum Readiness by Repository..... 92
Quantum Readiness by Language..... 93
Algorithm Usage Summary..... 94
Quantum Readiness by Algorithm Type..... 95
Code Scan Inventory..... 95
Quantum Trust Hub: Certificate Scan..... 101
Certificate Scan: Overview..... 101
Certificate Classification and PQC Risk Severity Assignment..... 101

Certificate Scan Dashboard....102
Quantum Readiness Score....103
Certificate Count....104
Quantum-Readiness Posture....105
High Risk Certificate Usage Report....105
PQC Risk Severity by Trust Hierarchy....107
Public Cert PQC Risk Severity....108
Internal Cert PQC Risk Severity....108
Algorithm Usage Summary....109
Key Usage Summary....109
EKU Usage Summary....110
Certificate Scan Inventory....110
Quantum Trust Hub: Configuration Scan....113
Quantum Readiness Score....115
Configuration Count....116
Quantum Readiness by Crypto Library....117
Quantum Readiness by Protocols....118
Risk Levels....119
Quantum Readiness by Key Exchange Usage in Cipher Suites....121
Quantum Readiness by Authentication in Cipher Suites....122
Applications Usage Summary....123
Quantum-Readiness Posture....124
Configuration Scan Inventory....125
Configuration Scan Inventory....125
Chapter 9. Configuring PQC Readiness/Post-Quantum Policies....129
Chapter 10. Quantum Solution Reference Guides....138
For Certificate Scan Outcome Analysis....138
For Configuration Scan Outcome Analysis....145
For Code Scan Outcome Analysis....175

Preface

Revision History

Revision	Description	Date
1.0	Initial draft of document for the release v2025.0.0.0	November 2025

About this Guide

This guide provides a comprehensive walkthrough of what is Post-Quantum Cryptography, the need for it, and AppviewX's implementation of its concepts to ensure that a seamless migration to a PQC ready cryptographic environment.

Audience

This guide is intended for decision-makers who need to understand quantum readiness posture and system and network administrators who track the corresponding metrics and plan strategic activities.

Third-Party Software Acknowledgments

This section serves as a placeholder to document the third-party components referenced in this guide, along with their associated trademark information.

Text Conventions

The following text conventions are used in this document:

Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1: Understanding PQC and the Need for It

Post-Quantum Cryptography, or PQC, refers to a new generation of cryptographic algorithms designed to withstand attacks from quantum computers.

Before we figure out what makes PQC the need of the hour in the world of information security, let's understand what quantum computers are and what exactly makes them a threat.

Quantum computers are an emerging computing system that use quantum mechanics principles like superposition and entanglement to process information. While classical computers use bits, which are strictly binary, quantum computers use qubits, which can exist in a superposition of 0 and 1 at the same time and hence, process several possibilities in parallel.

To simplify, a classical bit is like a coin lying flat on a table, so it's either heads or tails at one time, while a qubit is like a spinning coin, in a state of heads and tails simultaneously, depending on how you look at it.

Qubits allows quantum computers to solve problems exponentially faster, which includes breaking through the mathematical foundations of today's encryption standards, like RSA and ECC. Once powerful quantum computers exist, they could break most of the encryption that secures digital communication, financial systems, and sensitive data.

Now, here's the catch: quantum computers don't have to wait to 'exist' before they become a threat owing to the concept of 'harvest now, decrypt later'. This means that attackers can collect and store encrypted data today, even if they can’t break it yet. Once powerful quantum computers exist, the attackers will use them to break current encryption and read all this collected data.

This is why PQC and the quantum readiness of your cryptographic assets is urgent, and must be adopted before quantum computers are fully here.

PQC replaces today's vulnerable encryption algorithms with quantum-resistant ones. Adopting PQC ensures that data encrypted today remains secure in the future, protecting organizations from the 'harvest now, decrypt later' threat.

• AppViewX's Quantum Trust Hub Platform

AppViewX's Quantum Trust Hub Platform

AppViewX introduces the Quantum Trust Hub platform as an end-to-end solution to discover, classify, analyze, and assess cryptographic assets across applications, configurations, and codebases.

It helps organizations gain complete visibility into their cryptographic landscape, understand their exposure to quantum-related risks, and evaluate readiness for Post-Quantum Cryptography (PQC).Through automated discovery, deep analysis, and intelligent scoring, the platform delivers data-driven insights and expert recommendations to support a guided, monitored, and policy-aligned transition toward quantum-safe security.

The key features of this platform, along with its other capabilities (dashboards, inventory, policy creation) are explained in detail here.

Chapter 2: Prerequisites for Working with the AppViewX Config Scan and Code Scan Agents

AppViewX's PQC assessment tool is built with scanning capabilities to scan your entire cryptographic environment and generate reports that will help drive your organization's transition to a quantum-safe environment. The capabilities and working of this tool have been explained at length in this chapter.

This section of the guide focuses on configuring the following prerequisites before the PQC assessment tool begins working:

- Configure a service account to send reports to AppViewX.
- To be able to send reports via a cloud connector instance, configure a AppViewX Cloud Connector instance to send reports to the CC or a cloud DC

Configuring a Service Account to Send Reports to AppViewX

1. Login to AppViewX with your valid credentials.
2. Go to **Menu > Platform > Identity > Service Account**.
   The **Service Account** page is displayed, with the complete inventory of all existing service accounts.
3. From the menu bar, click **Add Account**.
   The **Service Account > Add** page is displayed.
4. In the **Account Information** section, enter/select the following service account details:

Feature	Description
***Name**	Name of service account.
Authenticate Externally	Enable the toggle to authenticate external service account using the client ID.
Client Id	This field is mandatory only when the **Authenticate Externally** toggle is enabled. Enter the external Client ID for authentication. When **Authenticate Externally** is disabled, the Client ID field will be auto-generated after registration.
Client Secret	This field is enabled only when **Authenticate Externally** toggle is disabled. The Client Secret is auto-generated after registration and is masked.

Feature	Description
	To view and copy the Client Secret, click the **Show/Hide** icon.
Description	Brief description of the service account
**: Mandatory fields*	

5. In the **Client Secret Settings** section, enter/select the following details:

Note: This section is displayed only if the **Authenticate Externally** toggle is disabled.

Feature	Description
***Client Secret Validity (in days)**	Specifies the number of days for which the generated client secret remains valid
Enable Secret Expiry Notification	When enabled, the system sends notifications to designated users or admins before a client secret expires. Alerts will be sent at intervals of 90, 60, 30, 7, and 1 day(s) before expiry and 1 day after expiry.
**: Mandatory fields*	

6. In the **Contact Information** section, enter the **Email Address** for creating the service account.
7. Click **Save**.

   The client ID and client secret are generated and displayed on the same page. These details have to be provided while running the code scan and the configuration scan agent.
8. Assign admin user groups for the created service account, or assign a user group with roles having Quantum Trust Hub ACF permissions.

   For instructions, see Enabling ACF Permissions for the Quantum Trust Hub.

Configuring a AppViewX Cloud Connector Instance to Send Reports

Prerequisites

- At least one Cloud Connector must be installed and running in the tenant.
- Cloud Connector version required: **2025.0.0.0**

- If the cloud connector is already installed but the version is not 2025.0.0.0:
  1. Upgrade the cloud connector to the version 2025.0.0.0.
  2. Verify the gateway is enabled following the instructions given here.
  3. If the gateway is not enabled, to enable the gateway, follow the instructions given here.
- If the cloud connector is installed for the version 2025.0.0.0:
  1. Verify the gateway is enabled following the instructions given here.
  2. If the gateway is not enabled, to enable the gateway, follow the instructions given here.
- Ensure that the Gateway pod is enabled in the Cloud Connector. If not, see Enabling the Gateway Pod with the HTTPS Profile in the Cloud Connector.
- Please provide the Cloud Connector hostname if you wish to communicate through the Cloud Connector while installing the Agent.

Checking if the Gateway with HTTPS Profile is Enabled in the Cloud Connector

1. Navigate to the Cloud Connector (CC) installation directory where the **install.sh** script is located.
2. Run the following command and verify that the output is similar to the one shown in the screenshot:

```
./deps/tools/k3s kubectl get svc -A | grep avx-mid-server-gateway-https
```

Enabling the Gateway Pod with the HTTPS Profile in the Cloud Connector

1. Navigate to the installation directory where the install.sh script is located in the cloud connector.
2. Set the following properties in the **deps/properties/appviewx.properties** file:

```
AUTO_ENROLL_ENABLED=true

ENABLE_HTTPS_PROFILE=true
```

3. Navigate back to install.sh script location in same CC installation folder and execute the following commands:

```
deps/tools/k3s kubectl scale deploy avx-mid-server-platform --replicas=0 -n cc
```

4. Wait until the platform pod is completely scaled down.
5. Restart the starter pod using below command to bring up both Gateway and Platform pods:

```
./avxctl restart starter
```

   Expected CC downtime: 2–5 minutes.
6. Verify the gateway is enabled following the instructions given here.

Chapter 3: Supported Ecosystems for PQC Assessment in AppViewX

Unified Scan

Scan Type	Deployment	OS Support	Server Support	Protocol Support	Crypto Assets-Discovery Scope
• Configuration Scan • Certificate Scan • Network Scan	Agent-based • Executable • Docker v20.10 and above	• Ubuntu 20.04 and above • RHEL 8 and above	Linux servers • Web servers (Apache, Nginx, Tomcat) • Mail servers (Postfix, Exim, Dovecot) • Application servers (JBoss, WildFly) • Database servers (MongoDB, MySQL, PostgreSQL)	• TLS/SSL • HTTPS • HTTP • FTPS • SMTP • IMAP • POP3 • LDAP	• Ciphersuites • Protocols • Algorithm • Certificates • Library

Code Scan

Deployment	OS Support	Language Support	Platform Vendor Support	Crypto Assets-Discovery Scope
Agent-based • Executable • Docker v20.10 and above	• Ubuntu 20.04 and above • RHEL 8 and above	• Java • Python • C • C++	• GitHub • GitLab • Azure DevOps • BitBucket • AWS CodeBuild • Local file system	• Crypto libraries (Java, Python) • Algoritms (Symmetric, Asymmetric, Key Derivation Function, Message Authentication Codes) • Certifiactes (CRT, CER, PEM)

Chapter 4: Configuring and Executing the AppViewX Config Scan Agent

A configuration scan examines system, network, and application settings to identify where and how cryptography is being used — specifically, which algorithms, protocols, and key strengths are configured in the environment.

AppViewX's Config Scan Agent scans the crypto libraries, protocols, cipher suites, and certificates bound to an application in your cryptographic environment, with the following three types of scans:

- Configuration scan (to identify vulnerabilities in protocols, cipher suites, crypto libraries, and certificates bound to an application running on a specific endpoint)
- Certificate scan (to identify vulnerabilities in certificates present in the file system)
- Network scan (to identify vulnerabilities in the protocols, cipher suites, and certificates bound to a specified IP address/range/subnet)

The scan results give you macro level visibility into the quantum readiness posture of your cryptographic configuration. The scan results are populated in the List of Scans in AppViewX's Quantum Trust Hub, and are displayed using interactive widgets on the Quantum Trust Hub dashboards.

In addition to this, the **Cryptographic Bill of Materials** (CBOM) generated as the output of the agent scan lists the vulnerabilities to help initiate your transition from a quantum vulnerable state to a quantum safe cryptographic environment.

The CBOM:

- Provides detailed insights into code, highlighting the line numbers where non-PQC compliant algorithms are used, along with the corresponding class names and algorithm names
- Includes remediation suggestions to help transition from non-PQC to PQC-compliant solutions.

The following two deployment models are available for installing the config scan agent:

- Linux-based
- Docker-based

The subsequent sections cover the installation instructions for each deployment type.

- Linux-based Installation of the Config Scan Agent
- Docker-based Installation of the Config Scan Agent

• Configuration File Prompts
• Additional Commands for Config Scan Agent Configuration

Linux-based Installation of the Config Scan Agent

Prerequisites

Supported operating systems

- Ubuntu: 20.04 and above (GLIBC >= 2.28)
- RHEL: 8 and above (GLIBC >= 2.28)

Privileges required

- sudo access (not needed if the user is a root user)
- Read access, for input files
- Execute access, for extracting service metadata
- Write access, for the output folder in which the scan reports must be saved

Installation Instructions

1. Login and download the latest Linux variant of the AppViewX Config Scan Agent from the release portal (link given below), in your Ubuntu or RHEL machine, as required.
   https://release.appviewx.com/downLoadArtifact?id=1733

   Note: This agent can be used to perform configuration scan, certificate scan, and network scan.

2. To verify that the tar file has not been tampered with and is secure for use.

   a. Download the signature file from the release portal, https://release.appviewx.com/downLoadArtifact?id=1730.

   b. Execute the following command:

   ```
   openssl cms -verify -in /path/to/SIG-file -inform DER -binary -noverify -content /path/to/tar-file
   ```

   Here:
   - **/path/to/SIG-file**: Signature file downloaded from release portal
   - **/path/to/tar-file**: Location of the tar file downloaded from the release portal

3. From the downloaded folder, execute the following command to untar the downloaded file:

```
tar -xvf config-scan-agent-linux-v1.0.0-2025.11.tar.gz
```

   The AppViewX Config Scan Agent, **config-scan-agent**, is extracted from the zipped file.

4. Save the AppViewX Config Scan Agent in all the endpoint machines where the configuration scan needs to be performed.
5. Verify if the execute permission has been assigned to the executable. If not, execute the following command:

```
chmod +x config-scan-agent
```

6. Create the configuration file required to run the AppViewX Config Scan Agent.

```
./config-scan-agent --create-config
```

   For additional options that you can use with the `./config-scan-agent` command, click here.

   Executing this command will Display a series of prompts to set up the configuration scan (as shown in the image below); your responses to these prompts will be entered in the configuration file.

```
A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness and by
default certificates will be uploaded directly in the platform's inventory in monitored status

Certificate Scan needed? (yes/no): yes

Enter the folder path containing the certificates intended for the Certificate Scan: /home/user/certs

Invalid folder path or folder path does not exist. Please provide a valid directory path.

Enter the folder path containing the certificates intended for the Certificate Scan: /home/

A Network Scan identifies active endpoints, protocols, and certificates in use to evaluate PQC readiness

Network Scan needed? (yes/no): yes

Enter IP range for network scan (single IP, comma-separated, range 192.168.x.x-192.168.x.x, subnet 192.168.x.x/y): 192.168.142.100-192.168.142
.250

Enter Ports range for network scan (single port, comma-separated, range 1-4000, default:443,8443): 1-32000

Enter number of IPs to scan per batch (maximum allowed value: 256, default: 256): 5

Uploading certificates adds them to the platform’s inventory in a monitored status for visibility and PQC readiness analysis

Do you want to upload certificates to AppViewX inventory? (yes/no): yes

Enter the path where reports should be saved: /home/appviewx/out

Configurations have been updated in '/home/appviewx/pqc/config.ini'.
```

Note: For a full list of the options available for configuring the AppViewX Config Scan Agent, execute the following command:

```
./config-scan-agent --help
```

7. Reply to the configuration prompts displayed.
   The prompts and their recommended responses are explained in detail here.

Important: You will also be required to enter your service account details as part of the configuration file set up. Ensure that the service account is created and you have the required details. For instructions, see Configuring a Service Account to Send Reports to AppViewX.

   The prompt responses will be used the following two files in the working directory of the host machine: **config.ini** (configuration file) and **secret.key** (file containing the secret key for the encryption and decryption)
8. To view the existing configuration, execute the following command:

```
./config-scan-agent --view-config <config_file_path> --secret-key <secret_key_path>
```

   Here:
   - **<config_file_path>**: Location of the configuration file
   - **<secret_key_path>**: Location of the secret key file
9. To perform the config scan, execute the following command:

```
./config-scan-agent --config <config_file_path> --secret-key <secret_key_path> --log-dir <log_directory>
```

   Here:

- **<config_file_path>**: Location of the configuration file
- **<secret_key_path>**: Location of the secret key file
- **<log_directory>**: Location of the file where the logs must be saved

After the scan is completed:

- Output reports (JSON/CSV/CycloneDX CBOM) will be saved in the output path provided in the configuration file.
- If enabled, the output reports will be sent to AppViewX and the corresponding data will be displayed in the List of Scans inventory in the Quantum Trust Hub.
- The configuration scan results will be displayed on the Configuration Scan Dashboard in the Quantum Trust Hub.
- Depending on the response to the corresponding prompt, certificates discovered as part of the scan will be uploaded to the AppViewX certificate inventory, in the **Monitored** state.
- Logs will be generated and saved in the working directory, by default.

  To save the logs in a custom log directory use the `<log_directory>`argument to specify the required location.

Additional Instructions

- To update the contents of the configuration file:
  1. Execute the following command:

     ```
     ./config-scan-agent --update-config <config_file_path> --secret-key <secret_key_path>
     ```

     The configuration prompts are displayed.

     ```
     Section: configuration_scan

     A Configuration Scan reviews Crypto libraries, protocols, cipher suites, service binding information's and certificates that are binded to an
     application to determine PQC readiness

     Config Scan needed? (yes/no) [To skip, press Enter]: no

     Section: certificate_scan

     A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness and by
     default certificates will be uploaded directly in the platform's inventory in monitored status

     Certificate Scan needed? (yes/no) [To skip, press Enter]: yes

     Folder path containing the certificates intended for the Certificate Scan [To skip, press Enter]: /home/appviewx/

     Section: network_scan

     A Network Scan identifies active endpoints, protocols, and certificates in use to evaluate PQC readiness

     Network Scan needed? (yes/no) [To skip, press Enter]: yes

     IP range for network scan (single IP, comma-separated, range 192.168.x.x-192.168.x.x, subnet 192.168.x.x/y) [To skip, press Enter]:

     Ports range for network scan (single port, comma-separated, range 1-4000, default: 443,8443) [To skip, press Enter]: 1-65535

     Number of IPs to scan per batch (maximum allowed value: 256, default: 256)) [To skip, press Enter]: 5

     Uploading certificates adds them to the platform’s inventory in a monitored status for visibility and PQC readiness analysis

     Upload certificates to AppViewX inventory for Network Scan? (yes/no) [To skip, press Enter]: no

     Section: output
     ```

2. Update the prompt responses as required.

   All updated values will overwrite the existing values

- To update only the service account details, execute the following command and update the prompt responses:

```
./config-scan-agent --update-service-acc  <config_file_path> --secret-key <secret_key_path>
```

- To set log levels, log directory, output directory use the options provided in the - -help option.

  For example:

  - `--log-level INFO ['DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL']`
  - `--log-dir /path/to/log/dir`
  - `--output /path/to/output/dir`

- To rotate the secret key (by default generated every time a new configuration file is generated) used for encryption and decryption of the service account credentials and the configuration file, execute the following command:

```
./config-scan-agent --rotate-key <secret_key_path> --config <config_file_path>
```

  Pass the `--config` argument to continue using the existing configuration file with the newly rotated secret key.

Docker-based Installation of the Config Scan Agent

Prerequisites

Supported operating systems

- Linux

Software Prerequisites

- Docker or Podman

System Utilities Required

- Linux command-line utilities (used for scanning libraries): ldd, lsof, ss, strings, which

Privileges required

- **--privileged**: Runs the container with extended privileges, allowing it to identify and scan host processes.
- **--cap-add CAP_NET_RAW**: Grants the container raw socket capabilities, required for port probing and TLS/SSL handshake operations.
- **--network host**: Shares the host's network stack with the container, enabling direct IP and port scanning.
- **--pid=host**: Shares the host's process ID namespace, allowing the container to access and identify running host processes.
- **sudo** privilege: Required on the host to access application metadata and perform privileged operations from within the container.
- Read access, for input files
- Execute access, for extracting service metadata
- Write access, for the output folder in which the scan reports must be saved

Note: The container connects to the host machine via SSH to conduct in-depth scanning and detect associated libraries and dynamic configuration details.

Installation Instructions

1. Login and download the latest Docker variant of the AppViewX Config Scan Agent from the release portal (link given below), in your Ubuntu or RHEL machine, as required.
   https://release.appviewx.com/downLoadArtifact?id=1731

   Note: This agent can be used to perform configuration scan, certificate scan, and network scan.

2. Save the downloaded folder in the machines where the config scan needs to be performed.
3. To verify that the tar file has not been tampered with and is secure for use.

   a. Download the signature file from the release portal, https://release.appviewx.com/downLoadArtifact?id=1732.

   b. Execute the following command:

   ```
   openssl cms -verify -in /path/to/SIG-file -inform DER -binary -noverify -content /path/to/tar-file
   ```

   Here:
   - **/path/to/SIG-file**: Signature file downloaded from release portal
   - **/path/to/tar-file**: Location of the tar file downloaded from the release portal

4. From the downloaded folder, extract the config scan docker agent and script from the tar.

```
tar -xvf  config-scan-agent-docker-v1.0.0-2025.11.tar.gz
```

5. To load the docker image, execute **one** of the following commands:

```
sudo docker load -i config-scan-docker-agent.tar.gz
```

OR

```
podman load -i config-scan-docker-agent.tar.xz
```

The docker image is loaded onto the system docker.

6. To verify if a docker image, with the name, exists in the system, execute **one** of the following commands:

```
sudo docker images
```

OR

```
podman images
```

7. To view the available options, execute the following script:

```
./run-config-scan-agent.sh --help --podman --non-sudo
```

In the above script:

- if Docker is used, delete the `--podman` argument
- Use the `--non-sudo` argument if Docker has to be run as a non-sudo user (by default, the Docker runs with sudo privilege)

8. Create the configuration file.

For Docker

```
./run-config-scan-agent.sh --create-config
```

For podman

```
./run-config-scan-agent.sh --create-config --podman --non-sudo
```

For additional options that you can use with the `./run-config-scan-agent` command, click here.

Executing this command will display a series of prompts to set up the configuration scan (as shown in the image below); your responses to these prompts will be entered in the configuration file.

```
Creating configuration for AVX Quantum Trust Hub ConfigScan Agent

Do you want to send reports to AppViewX (if AppViewX enabled)? (yes/no): yes

Enter the type of AppViewX deployment (on-premise/saas): saas

Is Cloud-Connector enabled? (yes/no): yes

Enter Cloud-Connector Hostname: cc-hostname

Enter Cloud-Connector Port (default: 30020):

Enter Client ID of Service Account: ***********************************

Enter Client Secret of Service Account: *******************************

A Configuration Scan reviews Crypto libraries, protocols, cipher suites, service binding information's and certificates that are binded to an
application to determine PQC readiness

Config Scan needed? (yes/no): yes

Uploading certificates adds them to the platform’s inventory in a monitored status for visibility and PQC readiness analysis

Do you want to upload certificates to AppViewX inventory? (yes/no): yes

A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness and by
default certificates will be uploaded directly in the platform's inventory in monitored status

Certificate Scan needed? (yes/no): yes

Enter the folder path containing the certificates intended for the Certificate Scan: /home/user/certs
```

```
A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness and by
default certificates will be uploaded directly in the platform's inventory in monitored status

Certificate Scan needed? (yes/no): yes

Enter the folder path containing the certificates intended for the Certificate Scan: /home/user/certs

Invalid folder path or folder path does not exist. Please provide a valid directory path.

Enter the folder path containing the certificates intended for the Certificate Scan: /home/

A Network Scan identifies active endpoints, protocols, and certificates in use to evaluate PQC readiness

Network Scan needed? (yes/no): yes

Enter IP range for network scan (single IP, comma-separated, range 192.168.x.x-192.168.x.x, subnet 192.168.x.x/y): 192.168.142.100-192.168.142
.250

Enter Ports range for network scan (single port, comma-separated, range 1-4000, default:443,8443): 1-32000

Enter number of IPs to scan per batch (maximum allowed value: 256, default: 256): 5

Uploading certificates adds them to the platform’s inventory in a monitored status for visibility and PQC readiness analysis

Do you want to upload certificates to AppViewX inventory? (yes/no): yes

Enter the path where reports should be saved: /home/appviewx/out

Configurations have been updated in '/home/appviewx/pqc/config.ini'.
```

9. Reply to the configuration prompts displayed.

   The prompts and their recommended responses are explained in detail here.

   > **Important:** You will also be required to enter your service account details as part of the configuration file set up. Ensure that the service account is created and you have the required details. For instructions, see Configuring a Service Account to Send Reports to AppViewX.

   The prompt responses will be used to create the following two files in the working directory of the host machine: **config.ini** (configuration file) and **secret.key** (key file that will be used to decrypt the configuration file).

10. To view the existing configuration, execute the following command:

    For Docker

```
./run-config-scan-agent.sh --view-config <absolute_config_file_path> --secret-key <absolute_secret_key_path>
```

For podman

```
./run-config-scan-agent.sh --view-config <absolute_config_file_path> --secret-key <absolute_secret_key_path> --podman --non-sudo
```

By default, contents of the configuration file in the working directory are displayed.

To view the contents of a configuration file saved in a custom location, replace the `<absolute_config_file_path>` argument with the required location.

If the secret key is stored in a custom location, replace the `<absolute_secret_key_path>` argument with the required location.

11. To perform the config scan, execute the following command:

For Docker

```
./run-config-scan-agent.sh --config <absolute_config_file_path> --secret-key <absolute_secret_key_path>
```

For podman

```
./run-config-scan-agent.sh --config <absolute_config_file_path> --secret-key <absolute_secret_key_path> --podman --non-sudo
```

By default, the configuration file and the secret key saved in the working directory are used for executing the configuration scan

If a configuration file saved in a custom location has to be used for the scan, replace the `<absolute_config_file_path>` argument with the required location.

If a secret key saved in a custom location has to be used for the scan, replace the `<absolute_secret_key_path>` argument with the required location.

After the scan is completed:

- Output reports (JSON/CSV/CycloneDX CBOM) will be saved in the output path provided in the configuration file.
- If enabled, the output reports will be sent to AppViewX and the corresponding data will be displayed in the List of Scans inventory in the Quantum Trust Hub.
- The configuration scan results will be displayed on the Configuration Scan Dashboard in the Quantum Trust Hub.
- Depending on the response to the corresponding prompt, certificates discovered as part of the scan will be uploaded to the AppViewX certificate inventory, in the **Monitored** state.
- Logs will be generated and saved in the working directory, by default.

  To save the logs in a custom log directory use the `<log_directory>`argument to specify the required location.

Additional Instructions

- To update the contents of the configuration file:
  1. Execute the following command:

     ```
     ./run-config-scan-agent.sh --update-config <abs_config_file_path> --secret-key <abs_secret_key_path> --podman --non-sudo
     ```

     The configuration prompts are displayed.

     ```
     Section: configuration_scan

     A Configuration Scan reviews Crypto libraries, protocols, cipher suites, service binding information's and certificates that are binded to an application to determine PQC readiness

     Config Scan needed? (yes/no) [To skip, press Enter]: no

     Section: certificate_scan

     A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness and by default certificates will be uploaded directly in the platform's inventory in monitored status

     Certificate Scan needed? (yes/no) [To skip, press Enter]: yes

     Folder path containing the certificates intended for the Certificate Scan [To skip, press Enter]: /home/appviewx/

     Section: network_scan

     A Network Scan identifies active endpoints, protocols, and certificates in use to evaluate PQC readiness

     Network Scan needed? (yes/no) [To skip, press Enter]: yes

     IP range for network scan (single IP, comma-separated, range 192.168.x.x-192.168.x.x, subnet 192.168.x.x/y) [To skip, press Enter]:

     Ports range for network scan (single port, comma-separated, range 1-4000, default: 443,8443) [To skip, press Enter]: 1-65535

     Number of IPs to scan per batch (maximum allowed value: 256, default: 256)) [To skip, press Enter]: 5

     Uploading certificates adds them to the platform’s inventory in a monitored status for visibility and PQC readiness analysis

     Upload certificates to AppViewX inventory for Network Scan? (yes/no) [To skip, press Enter]: no

     Section: output
     ```

  2. Update the prompt responses as required.

     All updated values will overwrite the existing values

- To update only the service account details, execute the following command and update the prompt responses:

  ```
  ./run-config-scan-agent.sh --update-service-acc <abs_config_file_path> --secret-key <abs_secret_key_path> --podman --non-sudo
  ```

- To set log levels, log directory, output directory use the options provided in the `--help` option.

  For example:

  - `--log-level INFO ['DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL']`
  - `--log-dir /path/to/log/dir`
  - `--output /path/to/output/dir`

- To rotate the secret key (by default generated every time a new configuration file is generated) used for encryption and decryption of the service account credentials and the configuration file, execute the following command:

```
./run-config-scan-agent.sh --rotate-key <abs_secret_key_path> --config <abs_config_file_path>
```

Configuration File Prompts

Prompt	Allowed values	Description
Do you want to send reports to AppViewX (if AppViewX enabled)?	• Yes • No	Enter **yes** if an AppViewX instance is present in the user premises and is needed to send the scan reports to the AppViewX Quantum Trust Hub platform.
Enter the Host Machine Username to run the docker container	String	**This is applicable only for Docker agents.** Enter the host machine username where the docker agent is to run. When prompted while the agent is running, enter the SSH password for this user.
Enter the type of AppViewX deployment	• on-premise • SaaS	Enter your AppViewX deployment type. **Note:** For the Managed K8s deployment, enter **saas** as the deployment type.
Enter AppViewX Hostname/IP Address	Valid hostname/IP address	This is applicable only for the on-premise deployment type. Enter the AppViewX instance hostname or the corresponding IP address.
Enter AppViewX Port	Any valid port number between 1 and 65535	Enter the port number for the AppViewX instance.

Prompt	Allowed values	Description
	Default: **31443**	If this value is not entered, port 31443 will be enabled by default.
Is Cloud-Connector enabled?	• Yes • No	**This prompt is applicable only for a SaaS deployment.** If the user has the Cloud-Connector installed in the environment and must send requests through the Cloud Connector, enter **Yes**.
Enter Cloud-Connector Hostname	String	Enter the hostname of the cloud connector.
Enter Cloud-Connector Port	Valid port for the AppViewX Cloud Connector Default: **30020**	Enter the cloud connector port number. By default, the port 30020 is enabled.
Enter Tenant Hostname	String	To directly send a request, without the cloud connector, enter the hostname of the SaaS tenant. For the Managed K8s deployment, enter the IP address extracted from the Web URL of your managed K8s environment.
Enter Client ID of Service Account	String	Enter the service account Client ID configured for the Configuration Scan. A user group with roles having Quantum Trust Hub permissions should be associated with this service account.
Enter Client Secret of Service Account	String	Enter the service account Client Secret configured for the Configuration Scan.

Prompt	Allowed values	Description
		A user group with roles having Quantum Trust Hub permissions should be associated with this service account.
Config Scan needed?	• Yes • No	Enter **yes** if you want to perform configuration scanning. A Configuration Scan reviews crypto libraries, protocols, cipher suites, service binding information and certificates that are binded to an application to determine the PQC readiness.
Do you want to upload certificates to AppViewX inventory?	• Yes • No	Enter **yes** if you want to upload the scan output reports to AppViewX. Uploading certificates adds them to the platform’s certificate inventory in a monitored status for visibility and PQC readiness analysis.
Certificate Scan needed?	• Yes • No	Enter **yes** if you want to scan the certificates associated with the application. A Certificate Scan inspects all certificates in use or available, based on the file path provided by the user, to assess PQC readiness. By default, certificates will be uploaded directly in the platform's certificate inventory in monitored status.
Enter the folder path containing the certificates intended for the Certificate Scan	String	Enter the path of the input folder to scan certificates and upload them to the inventory. Use **/** for a full file system scanning.
Network Scan needed?	• Yes • No	Enter **Yes** to include network scanning in the configuration scan.

Prompt	Allowed values	Description
		A Network Scan identifies active endpoints, protocols, and certificates in use to evaluate PQC readiness.
Enter IP range for network scan	• Single IP • Comma-separated range of IPs 192.168.x.x-192.168.x.x • subnet 192.168.x.x/y	Enter valid IP address/range/subnet details to perform SSL/TLS handshake and find the list of the associated protocols and ciphers.
Enter Ports range for network scan	Valid ports between 1 and 65535 **Default**: 443,8443 The values can be: • • A single port number • A comma-separated range between 1-4000	Enter the port/port range to perform SSL/TLS handshake. By default, ports 443 and 8443 will be scanned.
Enter number of IPs to scan per batch	Allowed range: 1 to 256 Default: **256**	Enter the number of IP addresses to be scanned in a single batch.
Enter the path where reports should be saved	String	Enter the output folder path in the file system where the scan reports need to be saved.

Additional Commands for Config Scan Agent Configuration

Option	Argument / Value	Description
`--version`	None	Displays the Config Scan Agent version **Example**: `./config-scan-agent --version`
`--license`	None	Displays the license information for the Config Scan Agent

Option	Argument / Value	Description
		Example: `./config-scan-agent --license`
`--config`	CONFIG_FILE_PATH	Specifies the location of the config file to be used for the scan. Use the `--secret-key` parameter along with the `--config` parameter to specify the secret key for decrypting the config file. **Example**: `./config-scan-agent --config /home/user/config.ini --secret-key /home/user/secret.key`
`--secret-key`	KEY_FILE_PATH	Specifies the path to the secret key file used for decrypting an encrypted config file
`--create-config`	None	Creates a config file in the current directory Also creates a secret key file in the same directory used for encryption/decryption of configuration file. **Example**: `./config-scan-agent --create-config`
`--update-config`	CONFIG_FILE_PATH	Interactively updates the configuration file at specified location Use the `--secret-key` parameter with the `--update-config` to specify the key location for encryption/decryption of the configuration file. **Example**: `./config-scan-agent --update-config /home/user/config.ini --secret-key /home/user/secret.key`
`--view-config`	CONFIG_FILE_PATH	Displays the decrypted contents of the config file at the specified location Use the `--secret-key` parameter with the `--view-config` to specify the key location for encryption/decryption of the configuration file. **Example**: `./config-scan-agent --view-config /home/user/config.ini --secret-key /home/user/secret.key`
`--rotate-key`	EXISTING_KEY_FILE_PATH	Rotates the secret key for the configuration file

Option	Argument / Value	Description
		Use the `--config` parameter with the `--rotate-key` to use the existing configuration file with the newly rotated key. The key will be rotated and stored in the same file specified. Use **--config** parameter along with **--rotate-key** to use the existing configuration file with the newly rotated key. The key will be rotated and stored in the same file specified. **Example**: `./config-scan-agent --config /home/user/config.ini --rotate-key /home/user/secret.key`
`--output`	OUTPUT_FOLDER_PATH	Specifies the output directory where the scan results will be stored **Example**: `./config-scan-agent --config /home/user/config.ini --secret-key /home/user/secret.key --output /home/user/output/`
`--log-dir`	LOG_FOLDER_PATH	Specifies the directory path to store the log files generated during scan agent execution **Example**: `./config-scan-agent --config /home/user/config.ini --secret-key /home/user/secret.key --log-dir /home/user/logs/`
`--log-level`	DEBUG, INFO, WARNING, ERROR, CRITICAL	Sets the verbosity level of logging, letting you control how much information the application logs and/or displays A lower verbosity level shows only critical errors, while higher levels include warnings, informational messages, and detailed debugging output. This helps users or developers adjust the amount of detail they see based on their troubleshooting or monitoring needs. **Default log level**: INFO **Example**: `./config-scan-agent --config /home/user/config.ini --secret-key /home/user/secret.key --log-level DEBUG`
`--log-retain`	NO_OF_LOGS	Specifies the count of logs to be maintained in the log folder **Default value**: 20

Option	Argument / Value	Description
		Example: `./config-scan-agent --config /home/user/config.ini --secret-key /home/user/secret.key --log-retain 30`
`--update-service-acc`	CONFIG_FILE_PATH	Updates the service account credentials stored in the configuration file Use the `--config` and `--key` parameters, along with the `--update-service-account` parameter to specify the configuration file and the key file locations. **Example**: `./config-scan-agent --update-service-acc /home/user/config.ini --secret-key /home/user/secret.key`

Chapter 5: Configuring and Executing the AppViewX Code Scan Agent

AppViewX's code scan agent integrates with your code repositories to scan code for quantum vulnerabilities. The Code Scan Agent is triggered in the CI/CD pipeline every time code changes are made in your repository and generates the following output artifacts:

- **Cryptographic Bill of Materials** (CBOM)
  - Provides detailed insights into code, highlighting the line numbers where non-PQC compliant algorithms are used, along with the corresponding class names and algorithm names
  - Includes remediation suggestions to help transition from non-PQC to PQC-compliant solutions
- **Static Analysis Results Interchange Format** (SARIF)
  - Provides a comprehensive view of the cryptographic compliance status across the codebase

Deployment Types

AppViewX offers versatility in its deployment, catering to diverse organizational needs and infrastructures. To ensure seamless integration and comprehensive assessment, two primary deployment models are available:

- Linux Executable

  Supported OS
  - Ubuntu 20.04 or above
  - RHEL 8 or above
  - (GLIBC version should be above 2.28)
- Docker

  Supported OS: Any operating systems that support Docker Engine v20.10 or later versions

AppViewX Code Scan Agent: Execution Flow

Executing the code scan agent is a three-step process, as outlined below:

1. Create the configuration file.
2. Integrate the AppViewX Code Scan Agent with the CI/CD pipeline of your code repository.
3. Execute the AppViewX Code Scan Agent.

Each of these steps is covered in detail in the subsequent chapters and sections.

• Creating the Configuration File for Agent Set Up
• Integrating the AppViewX Code Scan Agent with CI/CD Pipelines
• Executing the Code Scan Agent
• Sample YAML files for CI/CD Integration

Creating the Configuration File for Agent Set Up

• Additional Commands for Code Scan Agent Configuration
• Configuration File Prompts

Creating a Linux-based Configuration File

1. Login to the release portal https://release.appviewx.com/downLoadArtifact?id=1729 and download the latest Linux variant of the code scan agent, **code-scan-agent-linux-v1.0.0-2025.09.tar.gz**, in your Ubuntu or RHEL machines, as required.
2. To verify that the tar file has not been tampered with and is secure for use.

   a. Download the signature file from the release portal https://release.appviewx.com/downLoadArtifact?id=1730.

   b. Execute the following command:

   ```
   openssl cms -verify -in /path/to/SIG-file -inform DER -binary -noverify -content /path/to/tar-file
   ```

   Here:
   - **/path/to/SIG-file**: Signature file downloaded from release portal
   - **/path/to/tar-file**: Location of the tar file downloaded from the release portal

3. Untar the Code Scan Agent.

   ```
   tar -xvf code-scan-agent-linux-v1.0.0-2025.09.tar.gz
   ```

4. Assign execution permissions for the Code Scan Agent.

   ```
   chmod +x code-scan-agent
   ```

5. Create the configuration file and secret key needed for executing the code scan agent.

   a. Execute the following command to initiate the file creation process:

   ```
   ./code-scan-agent --create-config
   ```

During config file creation, you can provide **--config** and **--key** arguments to specify your configuration file and secret key location, else by default these will be created in relative path.

For additional options that you can use with the `./code-scan-agent` command, click here.

b. Reply to the configuration prompts displayed.
The prompts and their recommended responses are explained in detail here.

> **Important:** You will also be required to enter your service account details as part of the configuration file set up. Ensure that the service account is created and you have the required details. For instructions, see Configuring a Service Account to Send Reports to AppViewX.

> **Important:** To run the code scan agent in the local file system, for the configuration prompt, **Is this config file creation intended for CICD?**, your response must be **No**.

The **config.ini** file and the **secret.key** file will be created in the relative path. The secret key will be used to decrypt the configuration file while the scans are being executed.

Creating a Docker-based Configuration File

Prerequisites

- Set the required permissions for the configuration file creation.

For **Docker**, execute the following commands:

```
sudo mkdir /path/to/config-directory

sudo chmod 666 /path/to/config-directory
```

For **podman**, execute the following commands:

```
mkdir -p /path/to/config-directory

chown <username>:<group> /path/to/config-directory
```

Configuration File Creation Instructions

1. Login to the release portal https://release.appviewx.com/downLoadArtifact?id=1727 and download the latest Docker variant of the code scan agent, **code-scan-agent-docker-v1.0.0-2025.09.tar.gz**, in your Ubuntu or RHEL machines, as required.
2. Verify that the tar file has not been tampered with and is secure for use.

   ```
   openssl cms -verify -in /path/to/SIG-file -inform DER -binary -noverify -content /path/to/tar-file
   ```

   Here:
   - **/path/to/SIG-file**: Signature file downloaded from release portal
   - **/path/to/tar-file**: Location of the tar file downloaded from the release portal
3. Untar the Code Scan Agent.

   ```
   tar -xf code-scan-agent-docker-v1.0.0-2025.09.tar.xz
   ```
4. Load the docker agent.

   ```
   docker load -i code-scan-agent-image-v1.0.0.tar.gz
   ```
5. Create the configuration file and secret key needed for executing the code scan agent.

   a. Execute the following command to initiate the file creation process:
   [Adjust paths and usernames as needed for your environment.]

   For **Docker** [Use sudo if required by your system.]

   ```
   sudo docker run --rm -it \
     -v "/path/to/config-directory:/config" \
     code-scan-agent:v1.0.0 \
     --config /config/config.ini \
     --key /config/secret.key \
     --create-config
   ```

   Here, **/path/to/config-directory** is the absolute path of the existing directory where the configuration file must be created.

   For **podman**

   ```
   podman run --rm -it \
    -v "/path/to/config-directory:/config:Z" \
    code-scan-agent:v1.0.0 \
    --config /config/config.ini \
    --key /config/secret.key \
    --create-config
   ```

Here, **/path/to/config-directory** is the absolute path of the existing directory where the configuration file must be created.

The :Z option ensures proper SELinux labeling.

For additional options that you can use with the `./code-scan-agent` command, click here.

b. Reply to the configuration prompts displayed.
The prompts and their recommended responses are explained in detail here.

> **Important:** You will also be required to enter your service account details as part of the configuration file set up. Ensure that the service account is created and you have the required details. For instructions, see Configuring a Service Account to Send Reports to AppViewX.

> **Important:** To run the code scan agent in the local file system, for the configuration prompt, **Is this config file creation intended for CICD?**, your response must be **No**.

The **config.ini** file and the **secret.key** file will be created in the relative path. The secret key will be used to decrypt the configuration file while the scans are being executed.

Sample Configuration Files

For agent running in CICD using Cloud Connector for communication

```
[appviewx@ispc-cert-rhel-n3 code_scan]$ ./code-scan-agent --create-config
Creating config.ini file...

Is AppViewX enabled? (yes/no): yes

For Managed K8s deployment, specify saas as the deployment type
Enter the AppViewX deployment type (on-premise/saas) : saas

Do you want to send reports to AppViewX? (yes/no): yes

Is Cloud-Connector enabled? (yes/no): yes

Enter Cloud-Connector Hostname: cc_hostname

Enter Cloud-Connector Port (default: 30020): 30020

Enter Client ID of Service Account: ****

Enter Client Secret of Service Account: ****

Is this config file creation intended for CICD? (yes/no): yes

Config file updated in 'config.ini'.

Encryption key saved to secret.key. Keep it safe to decrypt the config file later.
[appviewx@ispc-cert-rhel-n3 code_scan]$
```

For agent running in CICD using Tenant Hostname for communication

```
[appviewx@ispc-cert-rhel-n3 code_scan]$ ./code-scan-agent --create-config
Creating config.ini file...

Is AppViewX enabled? (yes/no): yes

For Managed K8s deployment, specify saas as the deployment type
Enter the AppViewX deployment type (on-premise/saas) : saas

Do you want to send reports to AppViewX? (yes/no): yes

Is Cloud-Connector enabled? (yes/no): no

For Managed K8s deployment, enter the IP address extracted from the Web URL of your managed K8s environment
Enter the Tenant Hostname: tenant_hostname

Enter Client ID of Service Account: *****

Enter Client Secret of Service Account: *****

Is this config file creation intended for CICD? (yes/no): yes

Config file updated in 'config.ini'.

Encryption key saved to secret.key. Keep it safe to decrypt the config file later.
[appviewx@ispc-cert-rhel-n3 code_scan]$
```

For agent running in local file system in On-Premise environment

```
[appviewx@ispc-cert-rhel-n3 code_scan]$ ./code-scan-agent --create-config
Creating config.ini file...

Is AppViewX enabled? (yes/no): yes

For Managed K8s deployment, specify saas as the deployment type
Enter the AppViewX deployment type (on-premise/saas) : saas

Do you want to send reports to AppViewX? (yes/no): yes

Is Cloud-Connector enabled? (yes/no): yes

Enter Cloud-Connector Hostname: cc_hostname

Enter Cloud-Connector Port (default: 30020):

Enter Client ID of Service Account: ****

Enter Client Secret of Service Account: ****

Is this config file creation intended for CICD? (yes/no): no

Specify the folder path containing source code to scan : /home/appviewx/

Enter Repository Name (if the input folder does not contain a Git repository): repo1

Enter number of repositories to be analysed per batch (default 1):

Do you want to scan JAR files in the given input location? (yes/no): yes

Provide the directory path for storing code scan output : /home/appviewx/out

Config file updated in 'config.ini'.

Encryption key saved to secret.key. Keep it safe to decrypt the config file later.
[appviewx@ispc-cert-rhel-n3 code_scan]$
```

Additional Commands for Code Scan Agent Configuration

Option	Argument / Value	Description
`--version`	None	Displays the Code Scan Agent version
`--license`	None	Displays the Code Scan Agent license information
`--config`	CONFIG_FILE_PATH	Specifies the location of the config file to be used for the scan
`--secret-key`	KEY_FILE_PATH	Specifies the path to the secret key file to be used for decrypting an encrypted config file
`--create-config`	None	Creates a config file at the specified location

Option	Argument / Value	Description
		If a location is not specified, by default, the config is created in the current directory.
`--update-config`	CONFIG_FILE_PATH	Allows you to interactively update the config file The **--key** parameter must be used along with the **--update-config** parameter to specify the location of the config file.
`--view-config`	CONFIG_FILE_PATH	Allows you to view the decrypted contents of the config file, saved at the specified location The **--key** parameter must be used along with the **--update-config** parameter to specify the location of the config file.
`--rotate-key`	NEW_KEY_FILE_PATH (optional)	Rotates the secret key for the configuration file Optionally, you can specify a new key file path. To do this, use the **--config** and **--key** parameters, along with the **--rotate-key** parameter, to specify the config file and the old key location.
`--input-folder`	INPUT_FOLDER_PATH	Specifies the input directory containing the source code to be scanned
`--output-folder`	OUTPUT_FOLDER_PATH	Specifies the output directory where the scan results will be stored
`--log-dir`	LOG_FOLDER_PATH	Specifies the directory path to store log files generated during execution
`--log-level`	DEBUG, INFO, WARNING, ERROR, CRITICAL	Sets the verbosity level of logging, letting you control how much information the application logs and/or displays A lower verbosity level shows only critical errors, while higher levels include warnings, informational messages, and detailed debugging output. This helps users or developers adjust the amount of detail they see based on their troubleshooting or monitoring needs. Default log level: **INFO**

Option	Argument / Value	Description
`--log-retain`	NO_OF_LOGS	Specifies the count of logs to be maintained in the log folder **Default value: 20**
`--update-service-account`	None	Updates the service account credentials stored in the config file To do this, use the **--config** and **--key** parameters, along with the **--update-service-account** parameter to specify the config file and the key file location.

Additional Instructions

- To view the configuration file, execute the following command:

```
./code-scan-agent --view-config /path/to/config.ini --key /path/to/secret.key
```

  Here
  - **/path/to/config.ini**: path to the configuration file
  - **/path/to/secret.key**: path to the secret key to decrypt the configuration file
- To modify the existing configuration file, execute the following command

```
./code-scan-agent --update-config /path/to/config.ini --key /path/to/secret.key
```

  Here:
  - **/path/to/config.ini**: path to the configuration file
  - **/path/to/secret.key**: path to the secret key to decrypt the configuration file

  The existing configuration file will be displayed. For each configuration prompt, the existing response will be displayed. To update the response to a configuration prompt, type in your new response or press **Enter** to retain the existing value.

Configuration File Prompts

Question/Prompt	Allowed Values	Notes
Is AppViewX enabled?	• Yes • No	Enter **Yes** if AppViewX has been successfully installed.
Do you wish to send reports to AppViewX?	• Yes • No	Enter **Yes** to display the scan results in the Quantum Trust Hub dashboards.
Enter the type of AppViewX deployment	SaaS/On-Premise	Enter your AppViewX deployment type.  **Note:** For the Managed K8s deployment, enter **saas** as the deployment type.
Is Cloud-Connector enabled?	• Yes • No	This prompt is applicable only for a SaaS deployment. If the user has the Cloud-Connector installed in the environment and must send requests through the Cloud Connector, enter **Yes**.
Enter Tenant Hostname	String	Enter the tenant Hostname without the https:// prefix. For the Managed K8s deployment, enter the IP address extracted from the Web URL of your managed K8s environment.
Enter Cloud-Connector Hostname	String	Enter the Cloud Connector hostname without the https:// prefix.
Enter Cloud-Connector Port	Applicable port number	Enter the Cloud Connector port number. Default value: **30020**
Enter AppViewX Hostname/IP Address	String	Enter the hostname or the IP address without the https:// prefix and port number.
Enter AppViewX Port	Positive number	Enter the port number used by the AppViewX application. Default value: **31443**

Question/Prompt	Allowed Values	Notes
	without decimals	
Enter Client ID of Service Account	String	Enter the service account Client ID configured in AppViewX for the Code Scan. The user group with roles having Quantum Trust Hub permissions should be associated with this service account.
Enter Client Secret of Service Account	String	Enter the service account Client Secret configured in AppViewX for the Code Scan. The user group with roles having Quantum Trust Hub permissions should be associated with this service account.
Is this config file creation intended for CICD?	• Yes • No	Enter **Yes**, if the code scan agent will be executed in the CICD environment. Enter **No** if the code scan agent will be executed in the local environment apart from CICD runner machine
Specify the folder path containing source code to scan	String	Enter the absolute path of the input folder that needs to be scanned.
Enter Repository Name (if the input folder does not contain a Git repository)	String	Enter an alias repository name, if the input folder does not have any git repository. This will be shown as the repo name in Code Scan Inventory in Quantum Trust Hub.
Do you want to scan jar files?	• Yes • No	Enter **Yes** to include jar files in the code scan.
Provide the directory path for storing code scan output	String	Enter the absolute path of the output folder to store the scanned results and CSV reports.
Enter number of repositories to be	Positive number	Enter a batch size if the input folder has several cloned repositories and needs to send the results in batch.

Question/Prompt	Allowed Values	Notes
analysed per batch (default 1)	without decimals	For example, if your organization has 100 repositories, it is recommended to include 5 repositories per batch. Default number of repositories scanned per batch: **1**

Integrating the AppViewX Code Scan Agent with CI/CD Pipelines

Modern software development demands robust security practices that adapt to emerging threats. Continuous Integration and Continuous Deployment (CI/CD) pipelines play a critical role in ensuring code quality and security throughout the software lifecycle. The Code Scan Agent seamlessly integrates into CI/CD pipelines, enabling organizations to embed quantum-resilient cryptographic assessments into their development workflows.

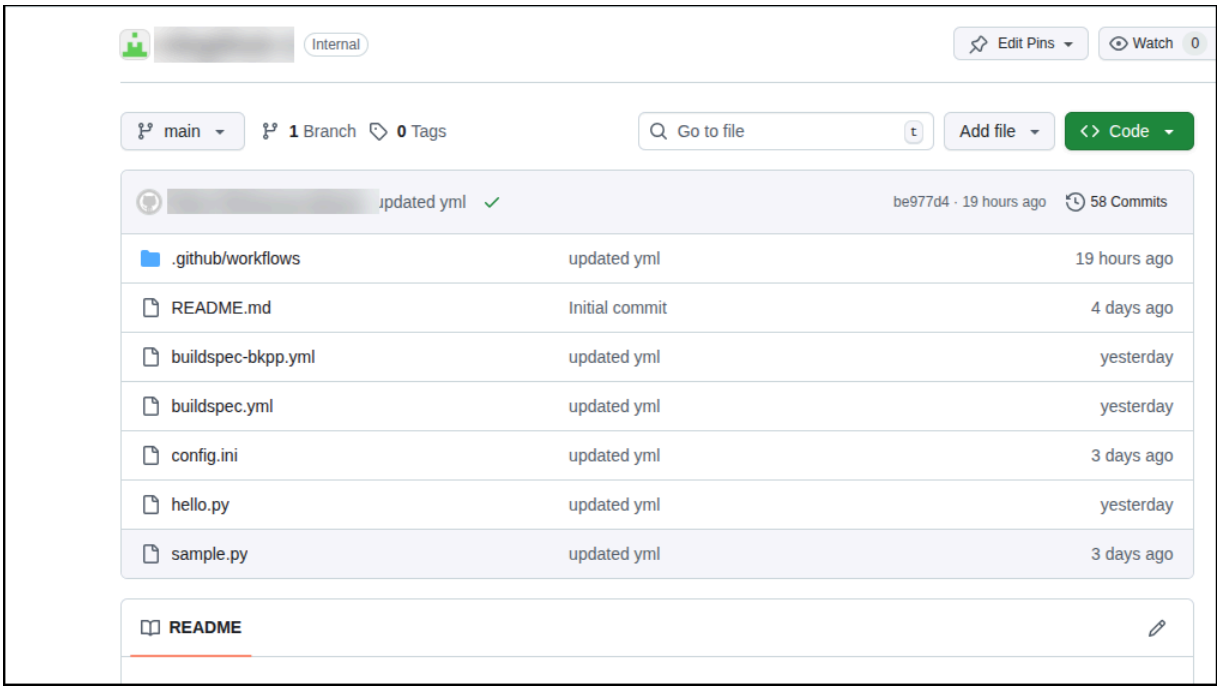
By automating cryptographic scans and remediation recommendations, the agent ensures that quantum-vulnerable algorithms are identified and addressed early in the development process.

Note: The integration steps are typically executed by a DevOps engineer.

GitHub Integration

To integrate the AppViewX Code Scan Agent with the GitHub pipeline:

1. To send the code scan reports to AppViewX, configure a service account in AppViewX.
2. Create a Linux executable configuration fileor create a Docker executable configuration file, as required.
3. To ensure the **config.ini** file is available during the pipeline process, save the **config.ini** file in the repository that needs to be scanned or in a centralized location from where the pipeline can access it for execution.



Following are a few examples of where the agent can be placed:

- In a repository that can be cloned during the pipeline process to retrieve the agent
- In Azure storage or an AWS S3 bucket, from where it can be pulled during the pipeline process

4. In an operating system environment supported by the Code Scan Agent, configure the GitHub runner for the repository that needs to be scanned.

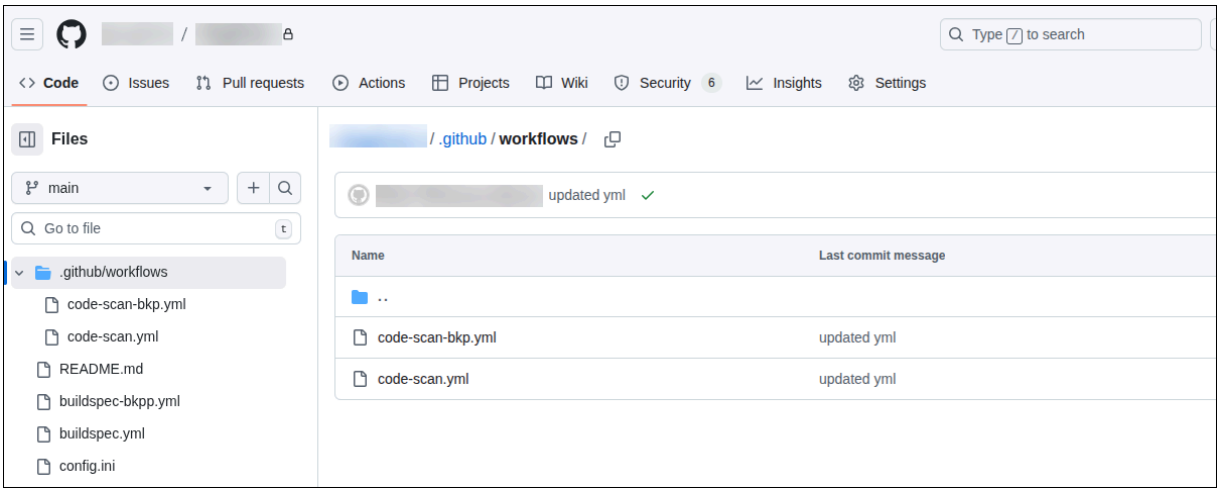
   The runner can be self-hosted or cloud-hosted, as long as it fulfils the prerequisites.

5. Ensure that:
   - The pipeline runner is running.
   - All runners meet the CICD executable prerequisites.
6. For the repository that needs to be scanned, configure the YAML file in **.github/workflows** for:
   - Cloning the repository including the **.git** folder

     The code scan agent picks the name, branch, and repository details from the **.git** folder.

     See the sample YAML file here.
   - Executing the AppViewX Code Scan Agent

     This sample file pulls the code scan agent executable from a S3 bucket. The **config.ini** file, for this execution, is placed in the same repository that has to be scanned. The secret key is passed a CICD variable for decrypting the config file.



7. To trigger the pipeline:

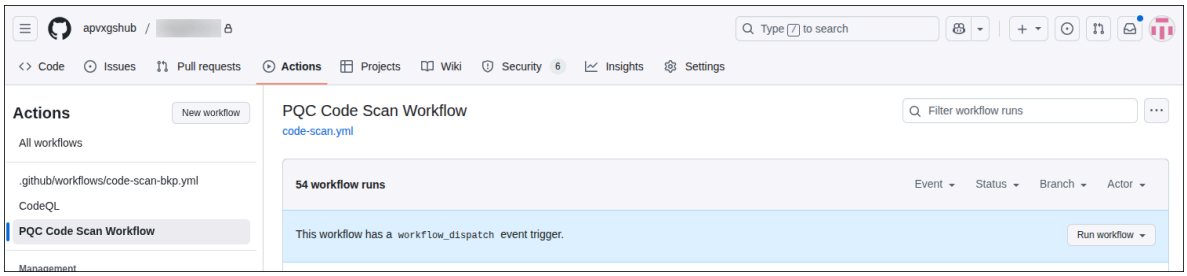
   a. Commit the code changes.

   OR

   a. To trigger the pipeline through **GitHub Actions**, go to the Git repository that has to be scanned.

   b. Go to **Actions** and select the required workflow name.

   c. Click **Run workflow**.



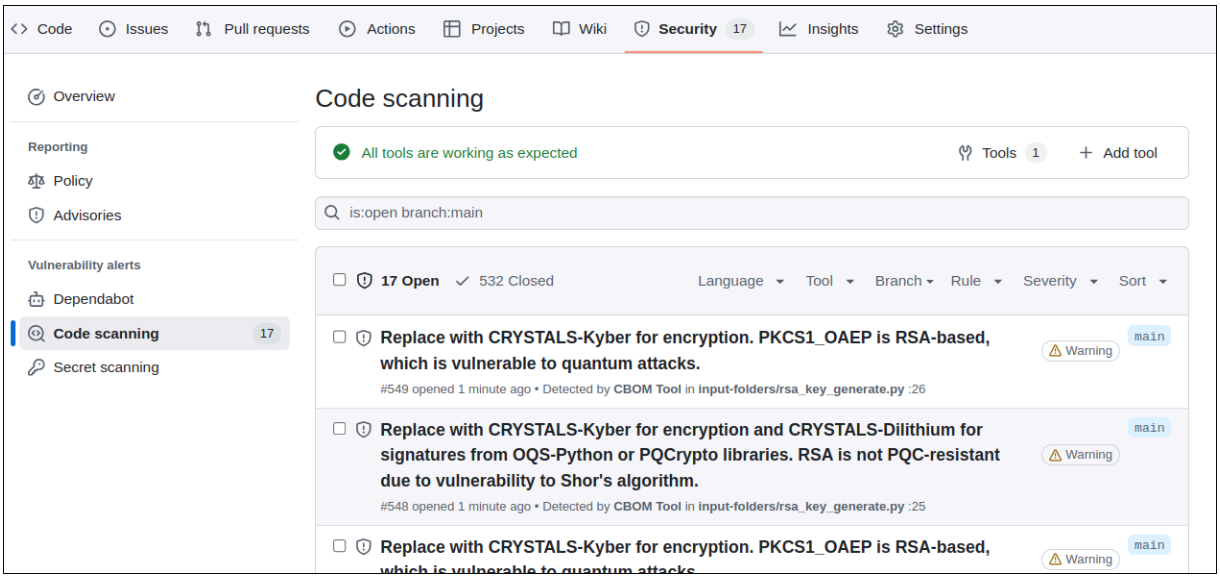
> **Note:** The **Run workflow** button, under GitHub **Actions**, is enabled only when the workflow includes the **workflow_dispatch** trigger.

Once the pipeline is triggered, the code scan agent is pulled and it begins scanning the repository.

You can view the status of the pipeline triggered under the **Actions** tab.

On completion of the scan, an output artifact containing the CycloneDX CBOM and the SARIF report is generated and made available in the **Artifacts** section.

8. To view the post-quantum vulnerabilities identified during the scan, go to the **Security** dashboard in your repository.



The SARIF results are also integrated in the **Security** tab. Within the **Security** tab, each non-compliance issue is clearly displayed, showing detailed information about the algorithm used, the affected code section, and its level of risk. The **Security** tab offers a direct and user-friendly interface for viewing these issues, making it easier for developers to identify and address them with appropriate remediation steps, guiding the development team towards a PQC-ready codebase.

If it has been enabled in the configuration file, the code scan results are also directly uploaded into the AppViewX platform and displayed on the Code Scan dashboard in the Quantum Trust Hub.

GitLab Integration

To integrate the AppViewX Code Scan Agent with the GitLab pipeline:

1. To send the code scan reports to AppViewX, configure a service account in AppViewX.
2. Create a Linux executable configuration fileor create a Docker executable configuration file, as required.
3. To ensure the **config.ini** file is available during the pipeline process, save the **config.ini** file in the repository that needs to be scanned or in a centralized location from where the pipeline can access the file for execution.
   Following are a few examples of where the agent can be placed:
   • In a repository that can be cloned during the pipeline process to retrieve the agent
   • In Azure storage or an AWS S3 bucket, from where it can be pulled during the pipeline process

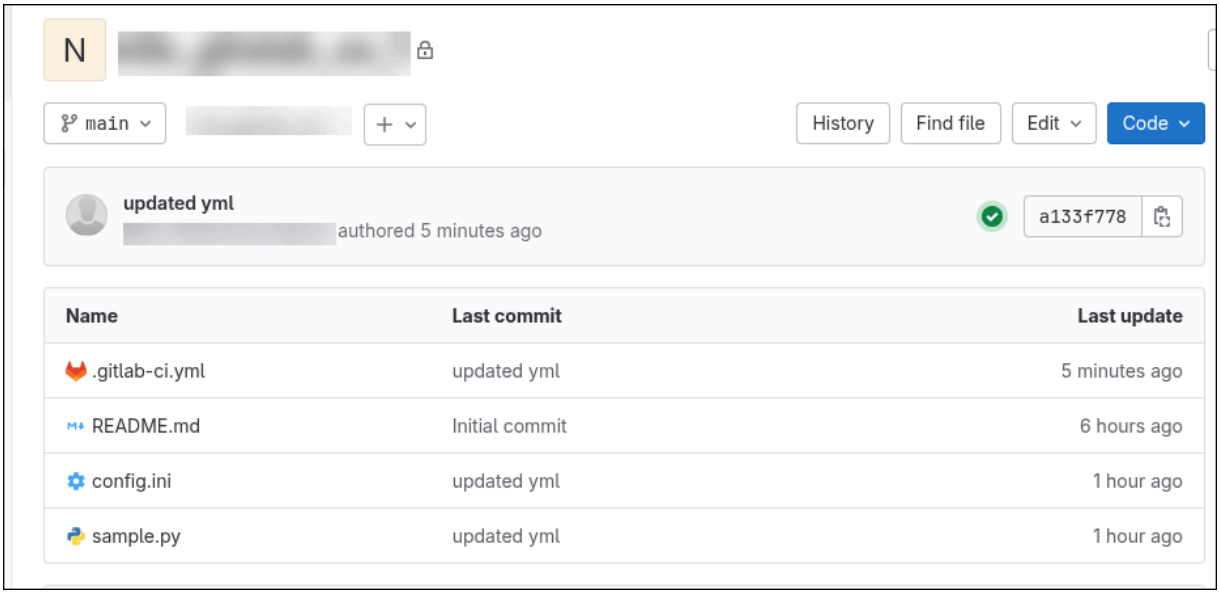
4. In an operating system environment supported by the Code Scan Agent, configure the GitLab runner for the repository that needs to be scanned.
   The runner can be self-hosted or cloud-hosted, as long as it fulfils the prerequisites.
5. For the repository that needs to be scanned, configure the YAML file in **.github/workflows** for:
   - Cloning the repository including the **.git** folder

     The code scan agent picks the name, branch, and repository details from the **.git** folder.
   - Executing the AppViewX Code Scan Agent

     This needs to be included in the YAML file to enable the agent to scan the source code and to provide the scan results.

   To create this YAML file, see the sample YAML file here.

   This sample file is configured to pull the code scan agent executable from a S3 bucket. The **config.ini** file, for this execution, is placed in the same repository that has to be scanned. The secret key is passed a CICD variable for decrypting the config file.

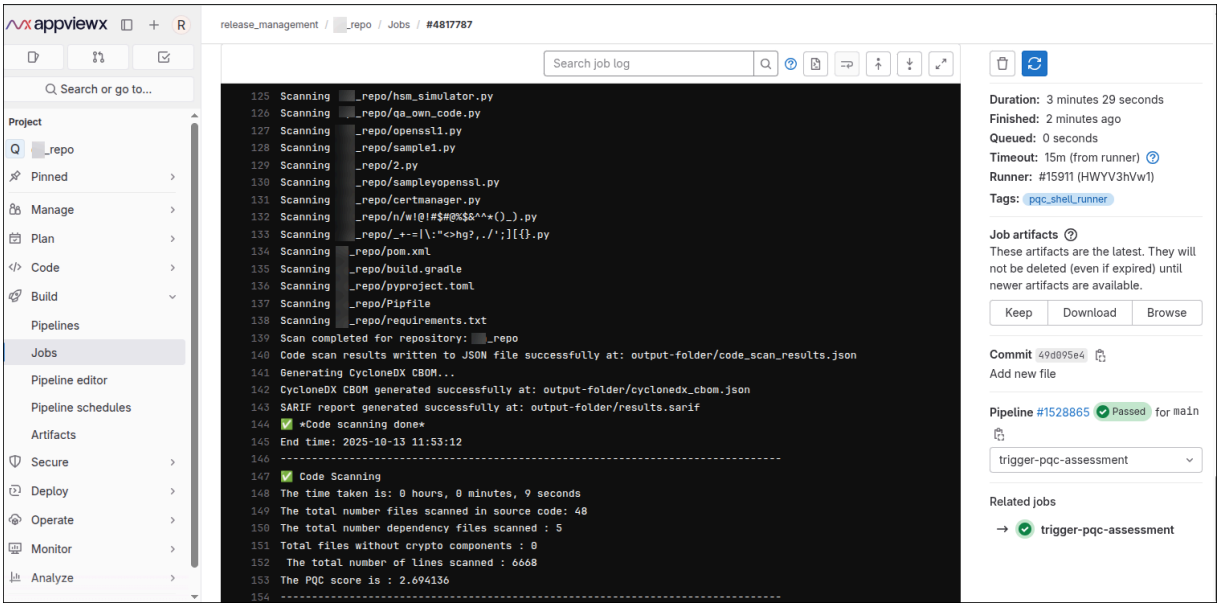


6. To trigger the pipeline:

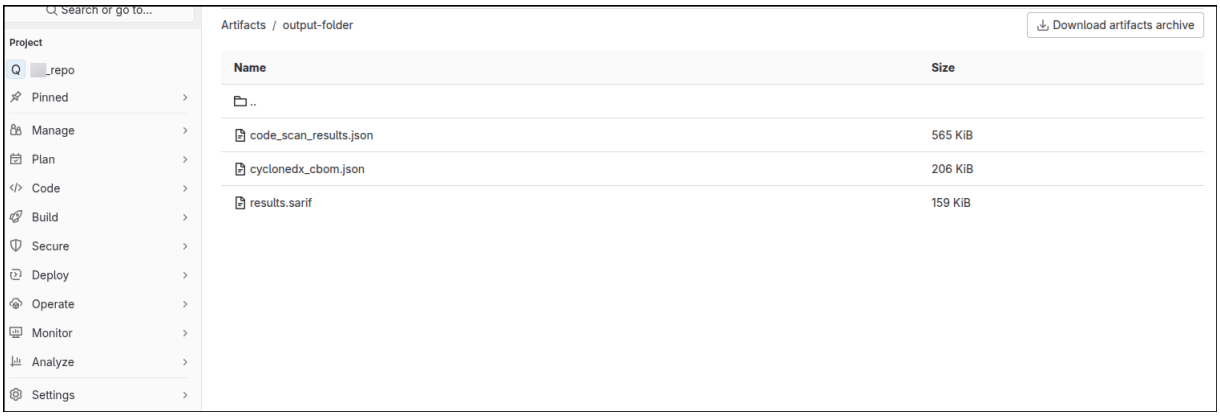
   a. Commit the code changes.
      OR

a. To trigger the pipeline from the GitLab console, go to the Git repository > **Build** > **Pipelines** > **Run Pipeline**.

Once the pipeline is triggered, the code scan agent is pulled and it begins scanning the repository.



On completion of the scan, an output artifact containing the CycloneDX CBOM and the SARIF report is generated and made available in the **Artifacts** section.



If it has been enabled in the configuration file, the code scan results are also directly uploaded into the AppViewX platform and displayed on the Code Scan dashboard in the Quantum Trust Hub.

BitBucket Integration

To integrate the AppViewX Code Scan Agent with the BitBucket pipeline:

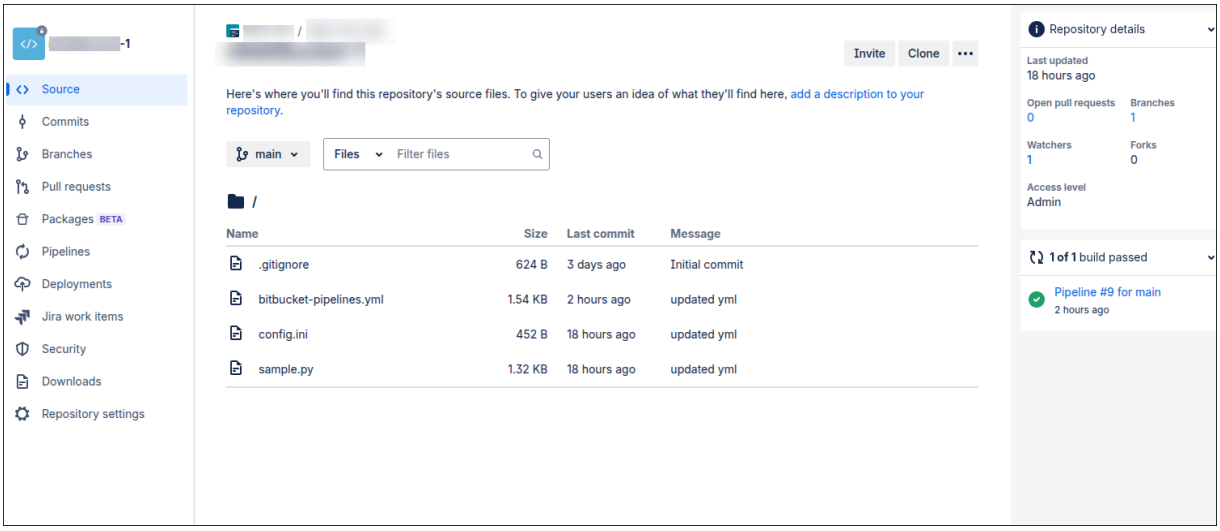
1. To send the code scan reports to AppViewX, configure a service account in AppViewX.
2. Create a Linux executable configuration fileor create a Docker executable configuration file, as required.
3. To ensure the **config.ini** file is available during the pipeline process, save the **config.ini** file in the repository that needs to be scanned or in a centralized location from where the pipeline can access the file for execution.
4. In an operating system environment supported by the Code Scan Agent, configure the BitBucket runner for the repository that needs to be scanned.
   The runner can be self-hosted or cloud-hosted, as long as it fulfils the prerequisites.
5. For the repository that needs to be scanned, configure the **.bitbucket-pipelines.yml** YAML file in **.github/workflows** for:
   - Cloning the repository, including the **.git** folder

     The code scan agent picks the name, branch, and repository details from the **.git** folder.

     See the sample YAML file here.
   - Executing the AppViewX Code Scan Agent

This needs to be included in the YAML file to enable the agent to scan the source code and to provide the scan results.

This sample file pulls the code scan agent executable from a S3 bucket. The **config.ini** file, for this execution, is placed in the same repository that has to be scanned. The secret key is passed a CICD variable for decrypting the config file.

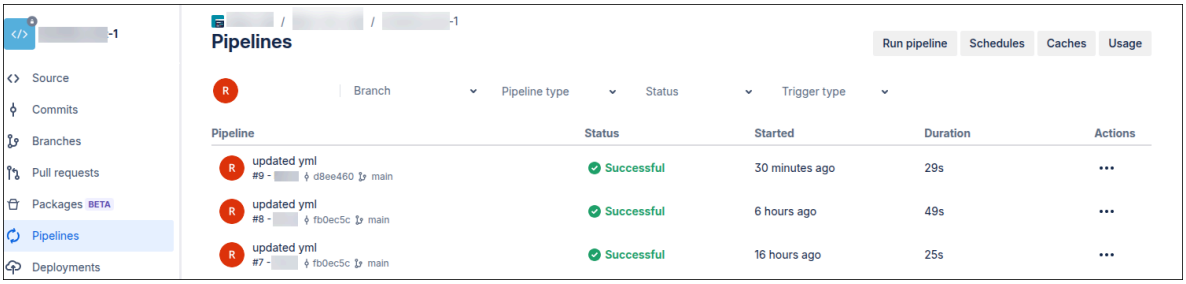


6. To trigger the pipeline:

   a. Commit the code changes.

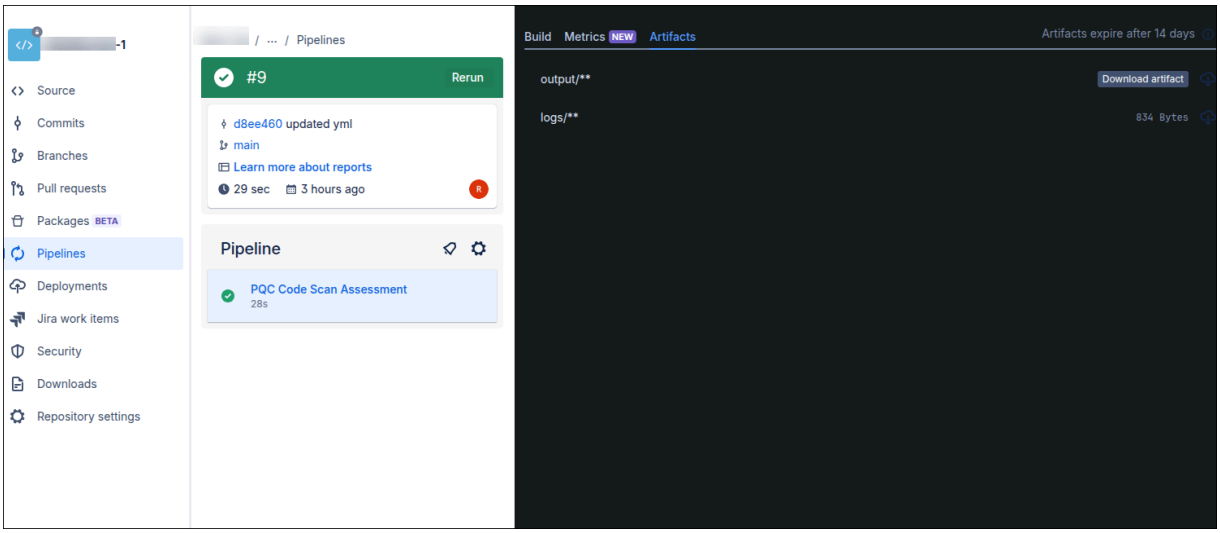
   OR

   a. To trigger the pipeline from the BitBucket console, go to the Git repository > **Pipelines**.
   b. Select the required pipeline and click **Run pipeline**.



Once the pipeline is triggered, the code scan agent is pulled and it begins scanning the repository.

On completion of the scan, an output artifact containing the CycloneDX CBOM and the SARIF report is generated and made available in the **Artifacts** section.



If it has been enabled in the configuration file, the code scan results are also directly uploaded into the AppViewX platform and displayed on the Code Scan dashboard in the Quantum Trust Hub.

Azure DevOps Integration

To integrate the AppViewX Code Scan Agent with the Azure DevOps pipeline:

1. To send the code scan reports to AppViewX, configure a service account in AppViewX.
2. Create a Linux executable configuration fileor create a Docker executable configuration file, as required.
3. To ensure the **config.ini** file is available during the pipeline process, save the **config.ini** file in the repository that needs to be scanned or in a centralized location from where the pipeline can access the file for execution.
4. In an operating system environment supported by the Code Scan Agent, configure the Azure DevOps runner for the repository that needs to be scanned.
   The runner can be self-hosted or cloud-hosted, as long as it fulfils the prerequisites.
5. For the repository that needs to be scanned, configure the **.azure-pipelines.yml** YAML file in **.github/workflows** for:
   - Cloning the repository, including the **.git** folder

     The code scan agent picks the name, branch, and repository details from the **.git** folder.

     See the sample YAML file here.
   - Executing the AppViewX Code Scan Agent

This needs to be included in the YAML file to enable the agent to scan the source code and to provide the scan results.

This sample file pulls the code scan agent executable from a S3 bucket. The **config.ini** file, for this execution, is placed in the same repository that has to be scanned. The secret key is passed a CICD variable for decrypting the config file.

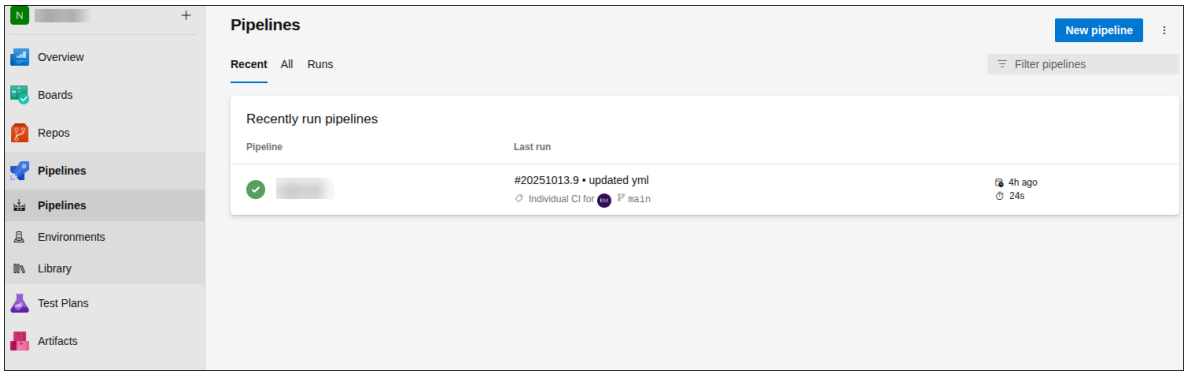
6. To trigger the pipeline:

a. Commit the code changes.

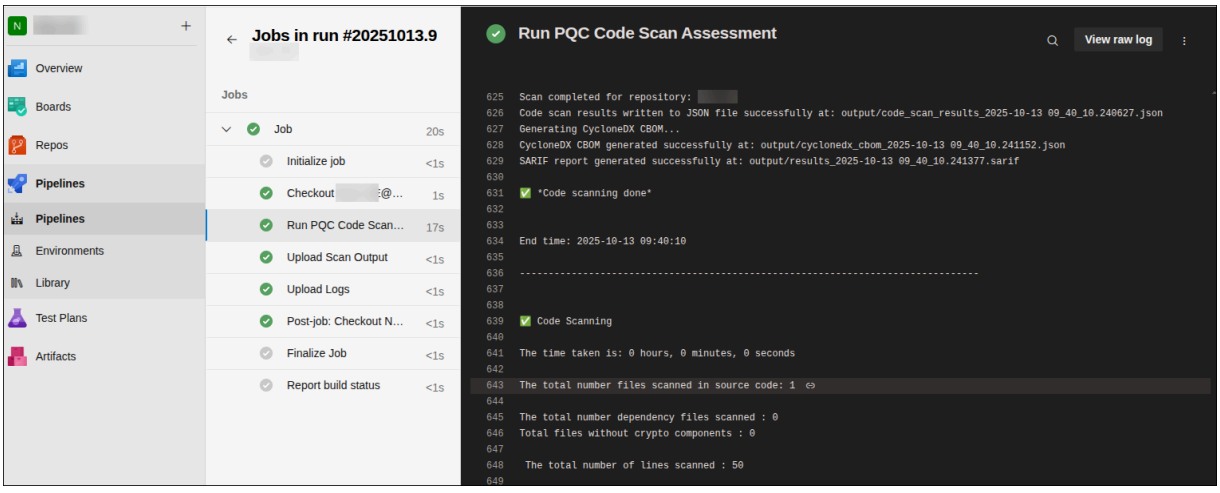
OR

a. To trigger the pipeline from the Azure console, go to the Git repository > **Pipelines**.

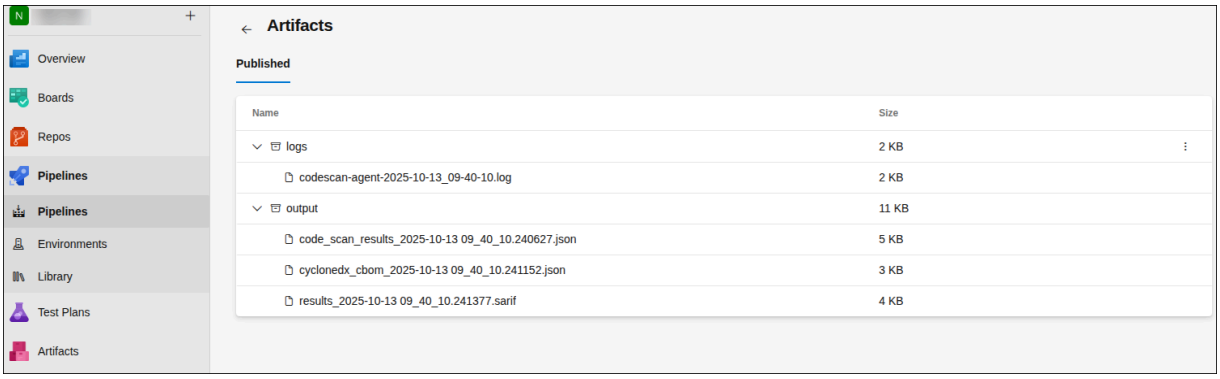
b. Select the required pipeline and click **Run pipeline**.



Once the pipeline is triggered, the code scan agent is pulled and it begins scanning the repository.



On completion of the scan, an output artifact containing the CycloneDX CBOM and the SARIF report is generated and made available in the **Artifacts** section.



If it has been enabled in the configuration file, the code scan results are also directly uploaded into the AppViewX platform and displayed on the Code Scan dashboard in the Quantum Trust Hub.

AWS Code Build Integration

To integrate the AppViewX Code Scan Agent with the AWS Code Build pipeline:

1. To send the code scan reports to AppViewX, configure a service account in AppViewX.
2. Create a Linux executable configuration fileor create a Docker executable configuration file, as required.
3. To ensure the **config.ini** file is available during the pipeline process, save the **config.ini** file in the repository that needs to be scanned or in a centralized location from where the pipeline can access the file for execution.
   Following are a few examples of where the agent can be placed:
   • In a repository that can then be cloned during the pipeline process to retrieve the agent
   • In Azure storage or an AWS S3 bucket, from where it can be pulled during the pipeline process
4. In an operating system environment supported by the Code Scan Agent, configure the AWS Code Build runner for the repository that needs to be scanned.
   The runner can be self-hosted or cloud-hosted, as long as it fulfils the prerequisites.
5. Ensure that:
   • The pipeline runner is running.
   • All runners meet the CICD executable prerequisites.
6. For the repository that needs to be scanned, configure the **buildspec.yml** YAML file in **.github/workflows** for:
   • Cloning the repository, including the **.git** folder

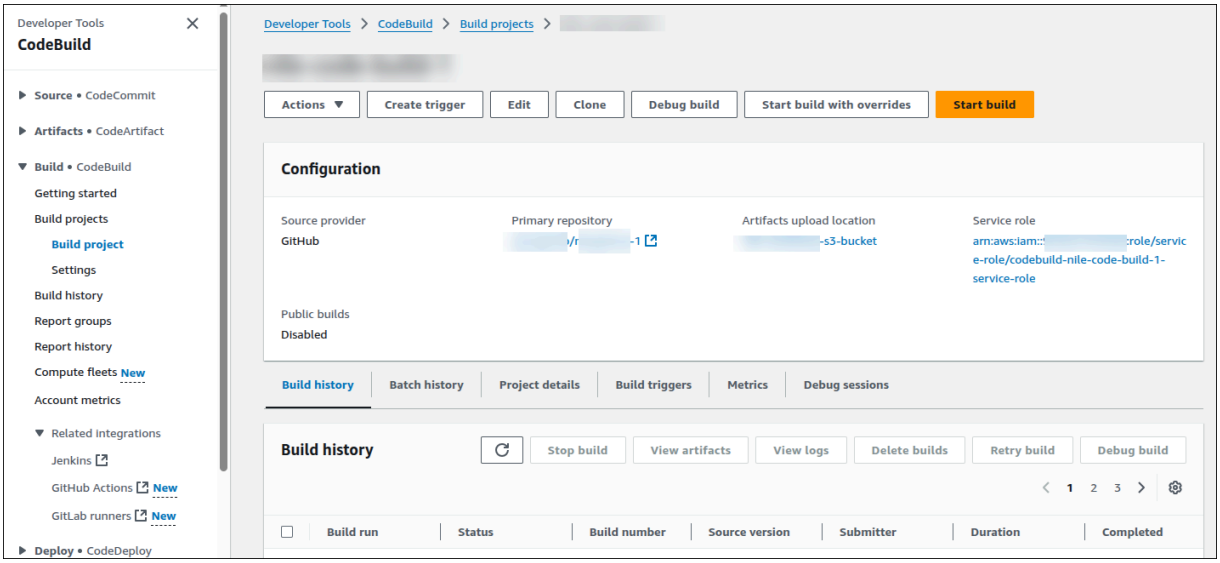
     The code scan agent picks the name, branch, and repository details from the **.git** folder.

     See the sample YAML file here.
   • Executing the AppViewX Code Scan Agent

This needs to be included in the YAML file to enable the agent to scan the source code and to provide the scan results.

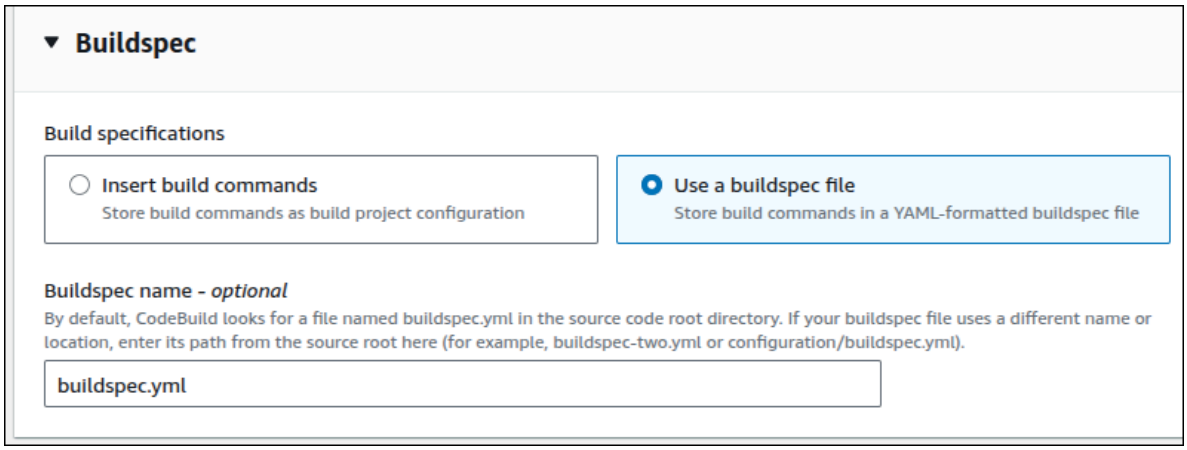
This sample file pulls the code scan agent executable from a S3 bucket. The **config.ini** file, for this execution, is placed in the same repository that has to be scanned. The secret key is passed a CICD variable for decrypting the config file.

7. Configure the AWS Code Build Project and Pipeline for the repository that needs to be scanned.



The source for this configuration can be any AWS-supporting CI/CD vendors.

8. From the **Operating System** dropdown list, select **Ubuntu** as the runner machine OS.
9. Under **Buildspec**:
   a. Select **Use a buildspec file**.



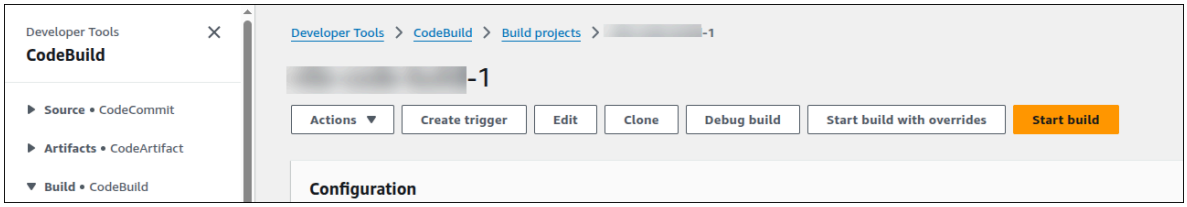
   b. In the **Buildspec name - *optional*** field, enter the YAML file name (**buildspec.yml**, for the purpose of this document).

10. To save the output artifacts generated by the code scan agent for later reference, under **Artifacts**, enter the details of the S3 bucket that can be used for uploading the artifacts.
11. To trigger the pipeline:

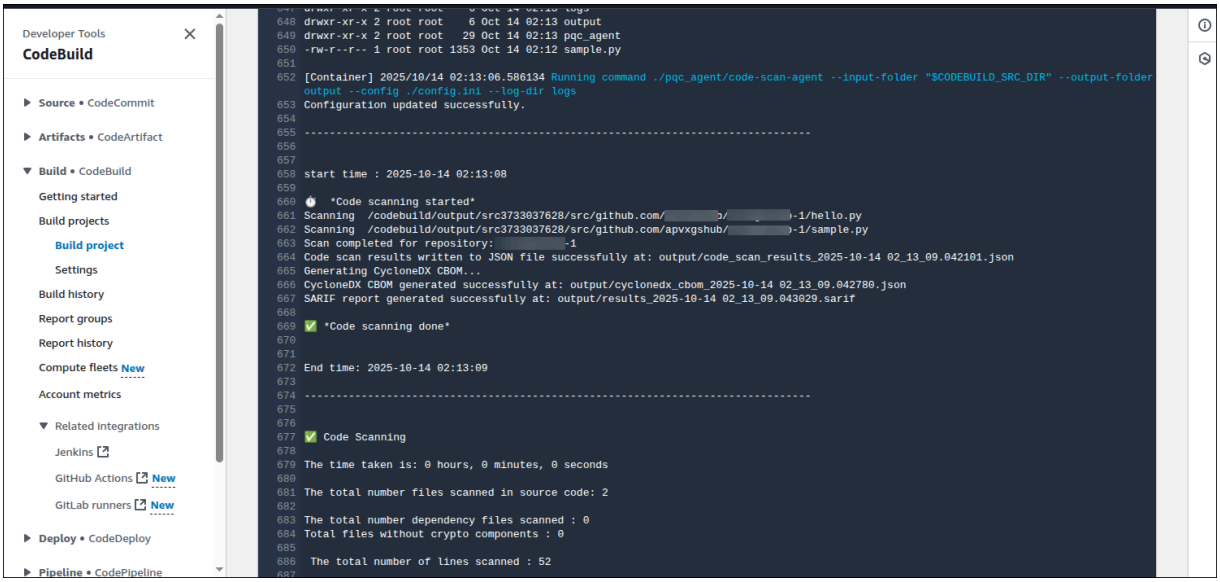
    a. Commit the code changes.

    OR

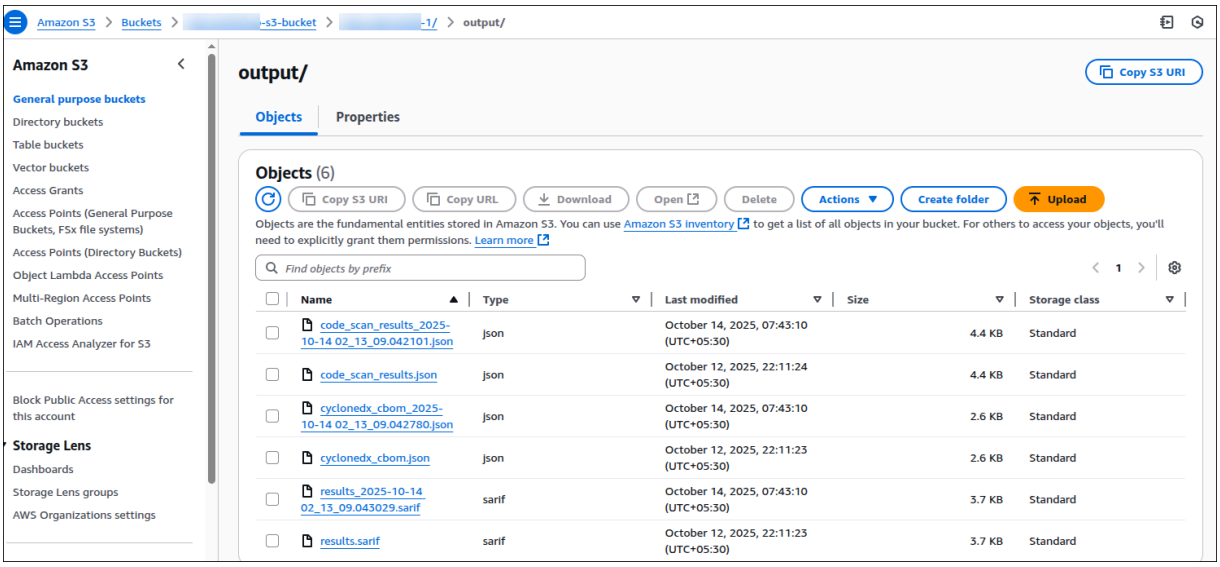
    a. To trigger the pipeline from the AWS console, in the **CodeBuild** section, for the repository to be scanned, click **Build Project**.



Once the pipeline is triggered, the code scan agent is pulled and it begins scanning the repository.



On completion of the scan, an output artifact containing the CycloneDX CBOM and the SARIF report is generated and made available in the **Artifacts** section.



If it has been enabled in the configuration file, the code scan results are also directly uploaded into the AppViewX platform and displayed on the Code Scan dashboard in the Quantum Trust Hub.

Executing the Code Scan Agent

Linux Executable-based Execution

Case 1: Executing the AppViewX Code Scan Agent via the CI/CD Pipeline

To execute the AppViewX Code Scan Agent in the CI/CD pipeline, execute the following code:

```
./code-scan-agent \
 --input-folder "/path/to/repository" \
 --output-folder "/path/to/output-folder" \
 --config "/path/to/config.ini" \
 --key "/path/to/secret.key" \
 --log-dir "/path/to/log-directory"
```

Here:

- **/path/to/repository**: absolute path of the input folder
- **/path/to/output folder**: absolute path of the output folder
- **/path/to/config.ini**: absolute path of the config.ini file
- **/path/to/secret.key**: absolute path of the key file that will be used to decrypt the config file
- **/path/to/log-directory**: absolute path of the log directory where logs needed to be stored

Case 2: Executing the AppViewX Code Scan Agent in the Local File System

Execute the following command in the server where the AppViewX Code Scan Agent and the configuration file are available:

```
./code-scan-agent --config /path/to/config.ini --key /path/to/secret.key
```

Here:

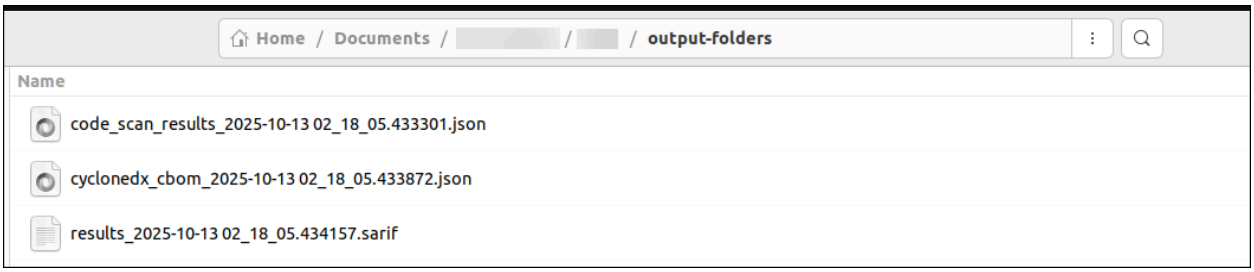
- **/path/to/config.ini**: absolute path of the config.ini file
- **/path/to/secret.key**: absolute path of the key file that will be used to decrypt the config file

Output (for both, Case 1 and Case 2)

The code scan agent will begin scanning the code in the input folder specified.

```
Scanning [redacted]/hsm_simulator.py
Scanning [redacted]/qa_own_code.py
Scanning [redacted]/openssl1.py
Scanning [redacted]/sample1.py
Scanning [redacted]/2.py
Scanning [redacted]/sampleyopenssl.py
Scanning [redacted]/certmanager.py
Scanning [redacted]/n/w!@!#$#@%$&^^*()_).py
Scanning [redacted]/_+-=|\:"<>hg?,./';][{}.py
Scanning [redacted]/pom.xml
Scanning [redacted]/build.gradle
Scanning [redacted]/pyproject.toml
Scanning [redacted]/Pipfile
Scanning [redacted]/requirements.txt
Scan completed for repository: [redacted]
Code scan results written to JSON file successfully at: output-folder/code_scan_results.json
Generating CycloneDX CBOM...
CycloneDX CBOM generated successfully at: output-folder/cyclonedx_cbom.json
SARIF report generated successfully at: output-folder/results.sarif
✅ *Code scanning done*
End time: 2025-10-12 11:09:07
--------------------------------------------------------------------------------
✅ Code Scanning
The time taken is: 0 hours, 0 minutes, 9 seconds
The total number files scanned in source code: 47
The total number dependency files scanned : 5
Total files without crypto components : 0
 The total number of lines scanned : 6631
The PQC score is : 2.694136
```

The CycloneDX CBOM output will be generated in the specified output folder. Along with CBOM, a SARIF report will be generated while running in CICD pipelines, which can be integrated with security dashboards like GitHub.



After the scan is completed, if your response to the question prompt Do you wish to send reports to AppViewX? was **Yes**, the scan results will be automatically uploaded to the AppViewX platform. These reports are displayed on the Code Scan dashboard and in the Code Scan Inventory in the Quantum Trust Hub.

Docker-based Agent Execution

Case 1: Executing the AppViewX Code Scan Agent via the CI/CD Pipeline

To execute the AppViewX Code Scan Agent in the CI/CD pipeline, execute the following code:

```
sudo docker run --rm \
 -v "/path/to/input-folder:/input-folder-name" \
 -v "/path/to/output-folder:/output-folder" \
 -v "/path/to/config.ini:/config.ini" \
 -v "/path/to/secret.key:/secret.key" \
 -v "/path/to/log folder:/logs" \
 code-scan-agent:v1.0.0 \
 --input-folder /input-folder-name \
 --output-folder /output-folder \
 --log-dir /logs \
 --config /config.ini \
 --key /secret.key
```

Here:

- **/path/to/input-folder**: absolute path of the input folder in which code scanning needs to be done
- **/path/to/output-folder**: absolute path of the output folder to store the scan results
- **/path/to/config.ini**: absolute path of the config.ini file,
- **/path/to/secret.key**: absolute path of the key file that will be used to decrypt the config file
- **/path/to/log-folder**: absolute path of the log directory where logs must be stored

Case 2: Executing the Code Scan Agent in the Local File System

To execute the Docker-based AppViewX Code Scan Agent in your local file system, execute the following code:

```
sudo docker run --rm \
 -v "/path/to/input-folder:/input-folder-name" \
 -v "/path/to/output-folder:/output-folder" \
 -v "/path/to/config.ini:/config.ini" \
 -v "/path/to/secret.key:/secret.key" \
 -v "/path/to/log-folder:/logs" \
 code-scan-agent:v1.0.0 \
 --log-dir /logs \
 --key /secret.key \
 --config /config.ini
```

Here:

- **/path/to/input-folder**: absolute path of the input folder in which code scanning needs to be done
- **/path/to/output-folder**: absolute path of the output folder to store the scan results
- **/path/to/config.ini**: absolute path of the config.ini file,
- **/path/to/secret.key**: absolute path of the key file that will be used to decrypt the config file
- **/path/to/log-folder**: absolute path of the log directory where logs must be stored

Tip: It is advised that you mount the input folder path with same name as the repository. For example, if the repository name is **repo_1** and the path is **/home/User/repo_1**, then to mount the input folder, execute the command: `-v “/home/user/repo_1:/repo_1”`.

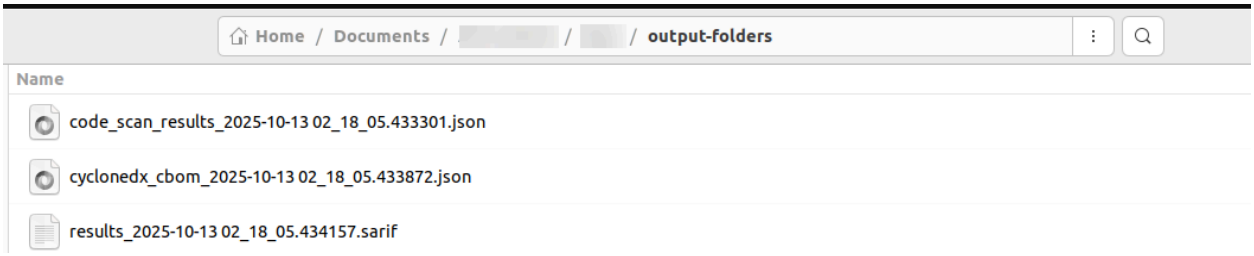
This ensures a valid relative path of the **filePath** from the repository name is shown in the inventory instead of showing the absolute path of the runner machine.

Output (for both, Case 1 and Case 2)

The code scan agent will begin scanning the code in the input folder mounted.

```
Scanning [redacted]/hsm_simulator.py
Scanning [redacted]/qa_own_code.py
Scanning [redacted]/openssl1.py
Scanning [redacted]/sample1.py
Scanning [redacted]/2.py
Scanning [redacted]/sampleyopenssl.py
Scanning [redacted]/certmanager.py
Scanning [redacted]/n/w!@!#$#@%$&^^*()_).py
Scanning [redacted]/_+-=|\:"<>hg?,./';][{}.py
Scanning [redacted]/pom.xml
Scanning [redacted]/build.gradle
Scanning [redacted]/pyproject.toml
Scanning [redacted]/Pipfile
Scanning [redacted]/requirements.txt
Scan completed for repository: [redacted]
Code scan results written to JSON file successfully at: output-folder/code_scan_results.json
Generating CycloneDX CBOM...
CycloneDX CBOM generated successfully at: output-folder/cyclonedx_cbom.json
SARIF report generated successfully at: output-folder/results.sarif
✅ *Code scanning done*
End time: 2025-10-12 11:09:07
--------------------------------------------------------------------------------
✅ Code Scanning
The time taken is: 0 hours, 0 minutes, 9 seconds
The total number files scanned in source code: 47
The total number dependency files scanned : 5
Total files without crypto components : 0
 The total number of lines scanned : 6631
The PQC score is : 2.694136
```

The CycloneDX CBOM output will be generated in the mentioned output folder. Along with CBOM, a SARIF report will be generated while running in CICD pipelines, which can be integrated with security dashboards like GitHub.



After the scan is completed, if your response to the question prompt Do you wish to send reports to AppViewX? was **Yes**, the scan results will be automatically uploaded to the AppViewX platform. These reports are displayed on the Code Scan dashboard and in the Code Scan Inventory in the Quantum Trust Hub.

Sample YAML files for CI/CD Integration

Note: Refer the sample YAML files given in this section and configure them accordingly for other CICD tools with the help of your Devops Engineers.

Sample Linux Executable YAML File

```
stages:
 - pqc-scan
variables:
 GIT_DEPTH: "0"  #Full clone depth to include .git directory.
pqc_code_scan:
 stage: pqc-scan
 image: ubuntu:latest
 before_script:
  # Prerequisite: Agent binary, `config.ini`, and `secret.key`
  # Prerequisite: Provide executable permission  to agent.
  - chmod +x ./code-scan-agent
  # Prerequisite: Prepare output + log directories.
  - mkdir -p scan-output scan-logs
 script:
  # Single mandatory execution step: runs the scan agent with required arguments. Provide the location of source code in argument --input-folder which was
checkout or cloned.
  - ./code-scan-agent \
      --input-folder /path/to/source-directory" \
      --output-folder scan-output \
      --config /path/to/config.ini \
      --log-dir scan-logs \
      --key s/path/to/secret.key
 artifacts:
  paths:
   - scan-output
   - scan-logs
  expire_in: 1 week
```

Sample Docker Executable YAML File

```
stages:
 - pqc-scan
```

```
variables:
  GIT_DEPTH: "0"  # Full clone depth to include .git directory.


pqc_code_scan:
  stage: pqc-scan
  image: ubuntu:latest
  before_script:
    # Prerequisite: Agent image, `config.ini`, and `secret.key` .
    - docker load -i code-scan-agent-image-v1.0.0.tar.gz
    # Prerequisite: Prepare output + log directories.
    - mkdir -p output-folder log-folder
  script:
    # Single mandatory execution step: runs the scan agent with required arguments. Provide the location of source code in argument --input-folder which was
 checked out or cloned.
    - |
      docker run --rm \
        -v "/path/to/input-folder:/input-folder-name" \
        -v "/path/to/output-folder:/output-folder" \
        -v "/path/to/config.ini:/config.ini" \
        -v "/path/to/secret.key:/secret.key" \
        -v "/path/to/log-folder:/logs" \
        code-scan-agent:v1.0.0 \
        --input-folder /input-folder-name \
        --output-folder /output-folder \
        --log-dir /logs \
        --config /config.ini \
        --key /secret.key
  artifacts:
    paths:
      - output-folder
      - log-folder
    expire_in: 1 week
```

Chapter 6: AppViewX Quantum Trust Hub User Guide for PQC Readiness

AppViewX's **Quantum Trust Hub** is a consolidated platform for tracking and managing your organization's PQC-readiness efforts. In the current implementation, the Quantum Trust Hub is made of dashboards, inventories, and a policy module.

The dashboards offer visibility into your organization's cryptographic usage, the vulnerabilities and corresponding severity levels in your cryptographic environment, and your overall progress towards PQC migration.

The inventories list all the cryptographic assets scanned for PQC-readiness and their quantum-safety status.

The policy module lets you define, manage, and enforce custom PQC policies to align with your organization's security goals.

Each module of the Quantum Trust Hub is explained in detail in the subsequent sections.

Key Features of the Quantum Trust Hub

- **Discover, classify, and manage cryptographic assets.**

  The platform automates the discovery of all cryptographic assets across your environment — including code, configuration files, applications, and endpoints.

  It identifies cryptographic elements such as algorithms, cipher suites, protocol versions, and key exchanges to build a comprehensive cryptographic inventory.

  The platform:
  - Detects and discovers both, direct and library-based cryptographic usage
  - Classifies assets as quantum-resistant, quantum-vulnerable, or hybrid, based on their cryptographic strength and quantum resilience
  - Enables you to monitor, track, and evaluate cryptographic assets across the organization from centralized interface

  This foundational visibility ensures that teams know **where and how cryptography is used**, setting the stage for targeted quantum readiness analysis.
- **Review cryptographic analysis reports for quantum-related vulnerabilities.**

  The platform:

- Does a deep analysis of your cryptographic environment and the included assets
- Auto-generates a Cryptographic Bill of Materials (CBOM) for consistent and repeated cryptographic assessments

  Each CBOM artifact contains a detailed breakdown of cryptographic components used within an application or configuration, enabling precise vulnerability analysis.

- Identifies quantum vulnerabilities using known algorithm weaknesses and key-size benchmarks
- Generates detailed reports outlining affected algorithms, risk levels, and exposure areas

This structured analysis helps organizations pinpoint cryptographic weaknesses that may be exploitable in a quantum era and supports data-driven risk evaluation.

- **Measure PQC-readiness for post-quantum adoption.**

  The platform:

  - Uses a policy-based scoring model tailored for your organization's needs
  - Assesses cryptographic components against customized security policies, taking into account algorithm strength, key usage, and protocol dependencies
  - Identifies weak, deprecated, or quantum-vulnerable algorithms
  - Displays readiness metrics to show how close each system or application is to PQC-readiness
  - Generates the Quantum Readiness Score, a quantitative indicator of your organization's readiness for post-quantum cryptography

  By consolidating technical analysis into a clear and measurable score, teams can track progress, prioritize upgrades, and plan their post-quantum migration strategy effectively.

- **Review and prioritize recommendations for quantum-safe transition.**

  While remediation actions remain manual and user-driven, the platform provides:

  - Contextual recommendations for replacing, upgrading, or deprecating vulnerable algorithms
  - Policy alignment suggestions to ensure all recommendations adhere to the enterprise security standards
  - Prioritization insights based on algorithm risk level, exposure scope, and readiness impact
  - Continuous improvement tracking through updated PQC scores and dashboards

  This recommendation-driven approach empowers users to make informed decisions without enforcing automatic changes, maintaining full operational control while advancing toward a quantum-safe environment.

- Licensing and Access Control for PQC in AppViewX
- Viewing the Quantum Trust Hub
- Understanding the Quantum Trust Hub Dashboards
- Organization Overview

• List of Scans
• Code Scan
• Configuration Scan
• Certificate Scan
• Quantum Trust Hub Inventory
• Quantum Trust Hub Policy

Licensing and Access Control for PQC in AppViewX

Licensing

PQC readiness in AppViewX is implemented and evaluated via the **Quantum Trust Hub**, the command center for tracking and managing your PQC readiness efforts.

The **Quantum Trust Hub** is built to work as an extension to AppViewX's flagship certificate lifecycle management product, **CERT+** (and not as a standalone product/module); to be able to explore the full capability of the **Quantum Trust Hub**, your CERT+ license must be upgraded as required.

For non-licensed users, the **Quantum Trust Hub** offers a focused read-only view of the existing certificate inventory's PQC readiness. For instructions to access this read-only view, click here.

For licensed users, the **Quantum Trust Hub** includes dashboards, inventories, and the ability to create custom quantum-safety policies, essentially everything you need to get a holistic view of your organization's PQC readiness and the required remediations. For instructions to view the **Quantum Trust Hub**, click here.

Enabling ACF Permissions for Quantum Trust Hub

Quantum Trust Hub uses role-based access control to ensure that only authorized users can access and perform operations such as running reports, managing inventory, setting policies, viewing alerts, and uploading to the library. ACF enablement details are provided in the corresponding sections.

To enable the ACF permission for the user roles to access the **Quantum Trust Hub**, follow these steps:

1. Go to **Platform** module **IDENTITY** > **Role**.
   You will be redirected to the Role page.
2. Click on the role name to enable the ACF permission.

   You will be redirected to the **Modify :: [RoleName]** page, with the **Information** tab open by default.

3. Switch to the **Authorized Functions** tab and expand the **Quantum Trust Hub** by clicking (Expand) icon. You can provide:
   a. Full access to Quantum Trust Hub feature by selecting corresponding checkbox which enables complete access to all features.
   b. Limited access to specific features by selecting the checkbox corresponding to features, such as Dashboard, Inventory, or Policy. This approach allows you to provide **limited access** without enabling the full Quantum Trust Hub.
4. Click **Save**.

Full List of ACF Permissions for RBAC

The Post-Quantum Cryptography platform implements Role Based Access Control (RBAC) to manage user permissions and access levels. Administrators can enable or disable specific features and operations based on user roles, ensuring appropriate access control and security governance across the organization.

Admin Control: All operations listed below are RBAC based and can be enabled or disabled by the administrator according to organizational requirements and user role definitions.

ACF Permission	Description
Quantum Trust Hub	
Publish Reports	Enable this option to allow report publishing on the platform. When enabled, users can generate and share reports across the organization.
Dashboard Access	
Dashboard	Access all organization reports, code, configuration, certificate scan results, and scan history from a single dashboard. This provides a centralized view of the organization's quantum readiness posture.
List of Scans	
List of Scan	Allows the user to access the inventory of scans conducted so far with their corresponding information.
View	Allows the user to view and access all scans that have been performed, including scan details, timestamps, and status.
Export	Enables the user to export the list of scans as CSV or Excel format for offline analysis and reporting.

ACF Permission	Description
Organization Report	
Organization Report	Enables viewing of organization-level reports and their related details, providing a comprehensive overview of the organization's cryptographic posture.
View	Permits the user to view organization reports within the dashboard, including PQC scores and aggregate metrics.
Refresh	Lets the user reload the organization report to show up-to-date information reflecting the latest scan results.
Code Scan Report	
Code Scan Report	Enables viewing of code scan reports with details on PQC readiness by class, method, and library.
View	Permits the user to view code scan reports within the dashboard, including detailed analysis of cryptographic implementations in source code.
Refresh	Lets the user reload the code scan report to show up-to-date information based on recent scans.
Configuration Scan Report	
Configuration Scan Report	Allows the user to review configuration scan results along with quantum-readiness status of system and application configurations.
View	Permits the user to view configuration scan reports within the dashboard, including protocol and algorithm settings.
Refresh	Lets the user reload the configuration report to show up-to-date information reflecting recent configuration changes.
Certificate Scan Report	
Certificate Scan Report	Allows the user to review detailed certificate scan results and their Post-Quantum status, including validity and cryptographic algorithm analysis.
View	Permits the user to view certificate scan reports within the dashboard, showing certificate details and quantum vulnerability assessment.
Refresh	Lets the user reload the certificate report to show up-to-date information based on the latest certificate scans.

ACF Permission	Description
Inventory Management **Note:** The inventory provides access to code and configuration data. To view certificate-related data, enable the required permissions from the **Certificate Inventory settings** outside this menu under **CERT+**.	
Code Inventory	
Code	Displays code inventory with details of repository, class, methods, library details, and their associated post-quantum readiness evaluation.
View	Enables viewing of code inventory along with quantum-readiness status across classes and methods, providing granular visibility into cryptographic implementations.
Export	Allows the user to export code inventory data in CSV or Excel format for reporting, analysis, and remediation planning.
Custom Library - Upload	Allows the user to upload custom libraries so that non-standard libraries can be detected and evaluated along with other standard libraries. This ensures comprehensive coverage of proprietary and third-party cryptographic implementations.
Configuration Inventory	
Configuration	Central repository for reviewing configuration scans of servers for post-quantum readiness evaluation, including protocol settings and cryptographic parameters.
View	Allows users to view configuration data with insights on post-quantum readiness, including detailed analysis of vulnerable settings.
Export	Allows the user to export configuration inventory data in CSV or Excel format for documentation and compliance reporting.
Policy Management	
Policy	Provides access to custom policies where organizations can define rules and override the default quantum status of algorithms or protocols based on application criticality, compliance requirements, or organizational standards.
View	Enables viewing of all configured policies and their related information, including policy rules, scope, and application.

ACF Permission	Description
Create	Allows the user to define and add new policies with required configurations, enabling customization of quantum readiness criteria.
Modify	Allows the user to modify existing policy configurations to adapt to changing security requirements or organizational priorities.
Delete	Allows the user to delete existing policies and their related information when they are no longer needed or applicable.

Viewing the Quantum Trust Hub

For Non-licensed Users (users with only a CERT+ license)

1. Login to AppViewX and go to **CERT+** > **Insights**.
   The **Insights** page is displayed.
2. From the **Explore Your PQC Readiness** banner, click **Go to Quantum Trust Hub**.
   The following read-only data is displayed:
   - Quantum Readiness Score
   - Quantum status wise count of certificates analyzed by the scan
   - Quantum readiness posture of your organization's cryptographic assets

For Licensed Users (users with the CERT+ license upgraded to include the Quantum Trust Hub)

To view the Quantum Trust Hub, login to AppViewX and go to **Menu** > **Quantum Trust Hub**.

By default, the **Quantum Trust Hub : List of Scans** page is displayed.

Understanding the Quantum Trust Hub Dashboards

The **Dashboard** module in the **Quantum Trust Hub** includes the following dashboards:

- **Organization Overview**
- **List of Scans**
- **Code Scan**
- **Configuration Scan**
- **Certificate Scan**

Organization Overview

The Post-Quantum Cryptography (PQC) Dashboard provides a comprehensive view of your organization's cryptographic posture and quantum readiness. After scanning your digital infrastructure, the dashboard consolidates critical security metrics to help you assess vulnerabilities and prepare for the quantum computing era.

A significant portion of the organization’s cryptography, configurations, and certificates are not quantum resistant, placing critical systems, sensitive data, and communications at high risk and necessitating immediate remediation. A summary combining the scan results from code, certificates, and configurations. Each section provides a brief overview and links to detailed views for the respective scan types.

Viewing the Organization Report

Prerequisite: Verify that your user role has the required **ACF permission** to view organization reports. To enable ACF permission, click here.

To view organization reports:

1. Go to **Menu > Quantum Trust Hub > Dashboard**.
   The **Quantum Trust Hub : Certificate Scan** page is displayed.
2. From the menu bar, select **Organization Overview**.
   The **Quantum Trust Hub : Organization Overview** page is displayed.

   The widgets displayed have been explained here.

Understanding the Organization OverviewDashboard

Quantum Readiness Score

The **PQC Score** provides an aggregate assessment of your cryptographic security posture against quantum threats. Higher scores indicate stronger quantum resistance across your infrastructure. Use this score to track progress over time and benchmark against industry standards.

PQC Score Breakdown

Score is calculated based on the Quantum readiness of the crypto categories identified across scans.

- Quantum Resistant Crypto Categories Identified = 1 Point
- Quantum Vulnerable Crypto Categories Identified = 0 Point

Total PQC Score = Total Quantum Resistant Crypto Categories identified / Total Crypto Categories Identified

(Higher scores indicate stronger overall organizational resilience against future quantum-based attacks.)

The threat level interpretation is therefore categorized as:

The threat level is displayed below the Gauge chart.

To read more on what the displayed threat level means and the recommended next steps, click **Know more** from the widget.

Cryptographic Asset Discovery Summary

Total Repositories Scanned

Total number of repositories analyzed during the scan process

Each repository is evaluated to identify cryptographic implementations, libraries, and algorithms used within the source code. The results contribute to determining the overall quantum readiness posture across the scanned repositories.

Total Cryptographic Dependencies

Represents the total count of cryptographic dependencies discovered during source code analysis. These include third-party libraries and packages that implement or utilize cryptographic functions such as. encryption, signing, or key exchange.

Total Endpoints Scanned

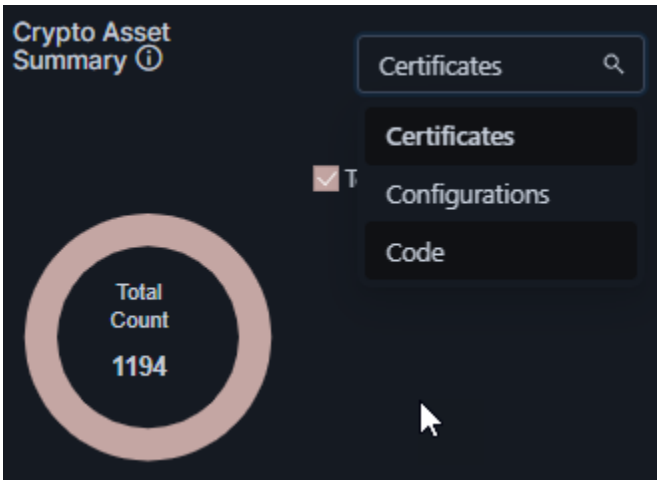
Indicates the total number of endpoints analyzed for cryptographic components and quantum readiness posture.

Total Applications Detected

Represents the total number of applications detected during the scan process. Each application is analyzed to identify embedded cryptographic components, libraries, and configurations that contribute to its overall quantum readiness posture.

These counts help you understand the scope of your cryptographic footprint and ensure complete coverage during scans.

Crypto Asset Summary



The **Crypto Asset Summary** widget provides a Visual breakdown of cryptographic assets across your organization, helping you understand where cryptography is used and identify potential areas of risk.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a specific crypto asset, from the following options:

Option	Description
Certificates	Displays cryptographic assets related to digital certificates, including types and algorithms. X.509 certificates and TLS/SSL implementations.
Configurations	Shows cryptographic elements within system or application configurations, like encryption settings and protocols.
Code	Represents cryptographic components found directly in the source code, such as implemented algorithms or cryptographic libraries.

High Risk Crypto Assets

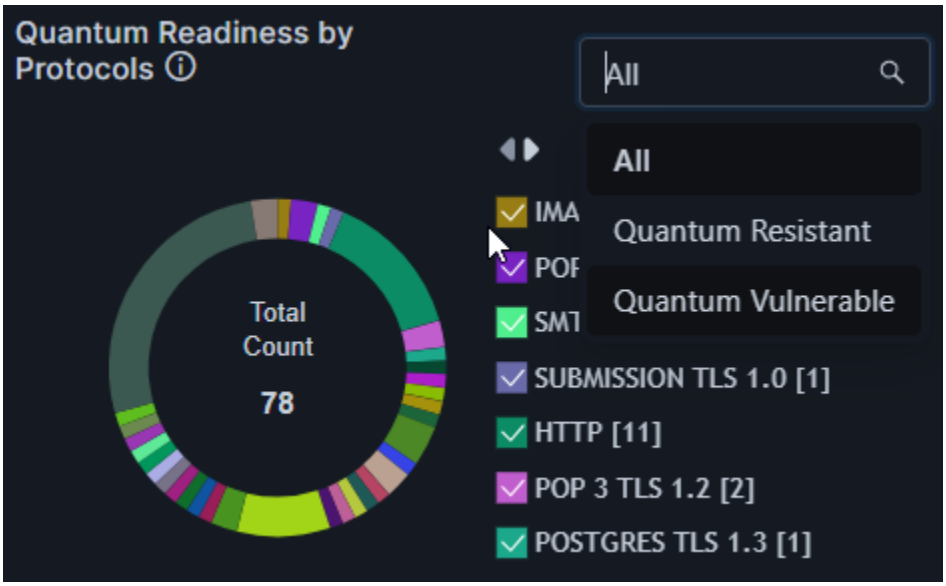
The **High Risk Crypto Assets** widget identifies cryptographic assets within your organization that use weak, outdated, or quantum vulnerable algorithms. These assets pose a significant security risk and require prioritization for remediation.

This widget visualizes the proportion of high-risk assets across different categories. It helps security teams focus on the most vulnerable areas of their cryptographic infrastructure.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a specific crypto asset, from the following options:

Option	Description
Certificates	Shows certificates using weak cryptographic standards, such as short key lengths or deprecated hashing/signature algorithms.
Configurations	Displays insecure cryptographic settings in system or application configurations, such as weak key exchange parameters or non-quantum resistant algorithms.
Code	Identifies hardcoded or referenced cryptographic algorithms in source code that are considered insecure or obsolete.

Quantum Readiness by Protocols

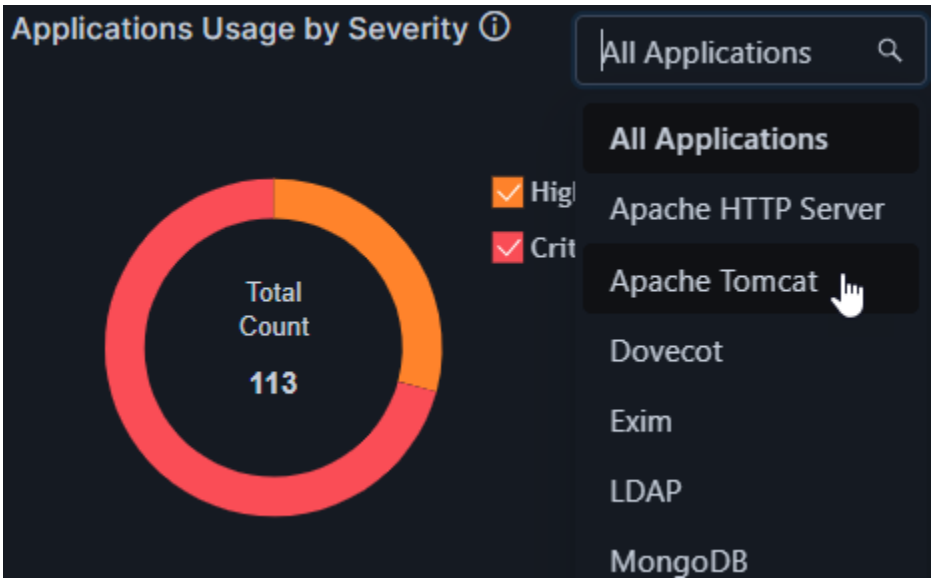


The **Quantum Readiness by Protocols** widget provides insight into the cryptographic protocols in use across your organization and evaluates their resilience against quantum computing threats. From the above widget, it can be inferred that the scan has detected 191 protocols currently in use. The interactive legends list all the protocols in use; the value in the square brackets indicates the frequency of use for that protocol.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a required quantum-safety status value, from the following options:

Option	Description
All	Displays the full distribution of cryptographic protocols detected across the environment, both quantum resistant and quantum vulnerable.
Quantum Resistant	Filters and displays only those protocols that are designed to resist quantum attacks (e.g., using post-quantum algorithms or strong modern standards).
Quantum Vulnerable	Filters and displays protocols that rely on cryptographic mechanisms known to be breakable by quantum computers (e.g., RSA, ECC < P-256, DH ≤ 2048-bit).

Applications Usage by Severity



The **Applications Usage by Severity** widget categorizes applications based on the severity of their cryptographic vulnerabilities across your organization's applications. This chart helps prioritize risk mitigation efforts based on the criticality of the issues identified.

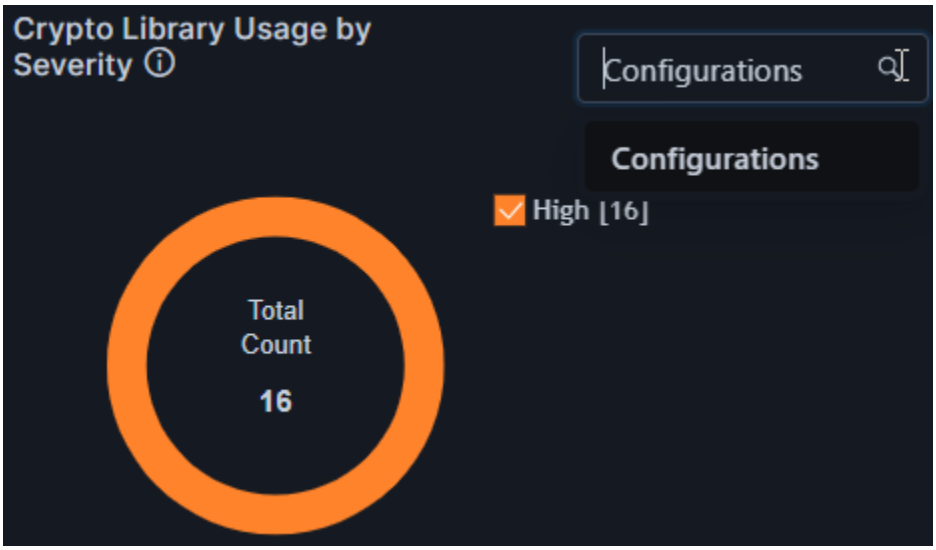
The above widget shows the number of applications currently in use, as detected by the scan. The interactive legend lists the severity status and the count of applications per severity status.

Severity levels typically include:

- **Critical:** High-impact vulnerabilities that pose an immediate threat to data security and require urgent remediation.
- **High:** Serious weaknesses that could be exploited but may require more effort or specific conditions.
- **Medium:** Moderate risks that should be addressed but are not immediately exploitable.
- **Low:** Minor issues or outdated practices that may not pose immediate risk but still warrant review.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a specific application.

Crypto Library Usage by Severity



The **Crypto Library Usage by Severity** widget provides a visual breakdown of cryptographic libraries used in applications that are consumed across your organization, categorized by the severity of vulnerabilities associated with them. This helps you identify which libraries pose the highest cryptographic risk and in which parts of the environment they are used.

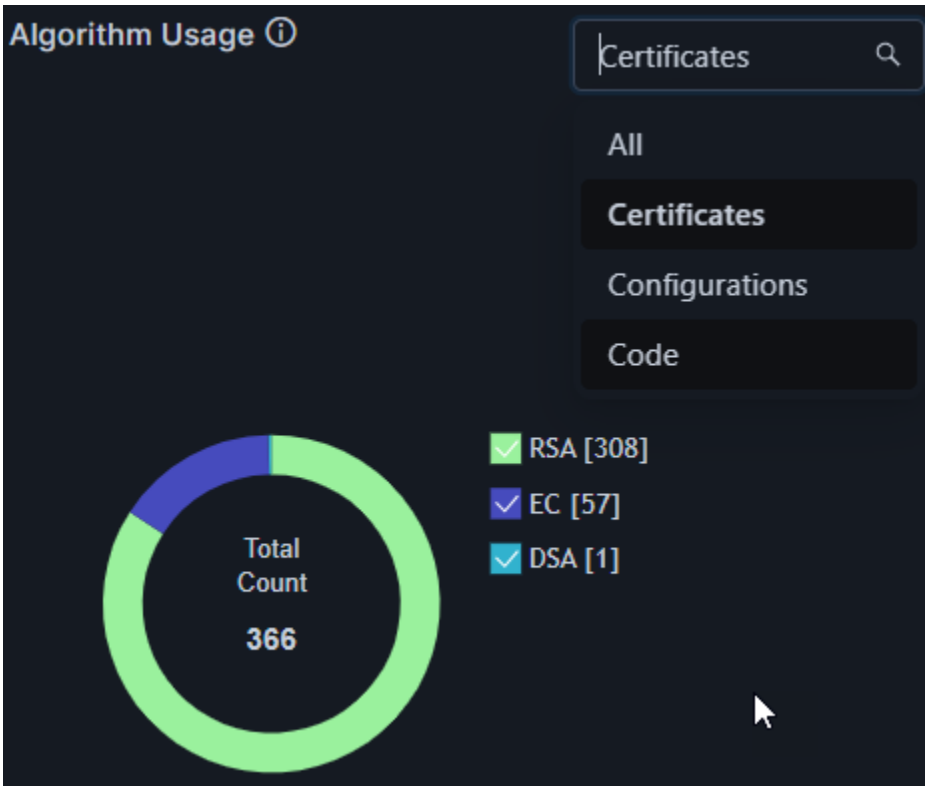
The above widget shows the total number of libraries in use. The interactive legend lists the severity levels and the count of applications per severity status.

Severity Levels typically include:

- **Critical:** High-impact vulnerabilities that pose an immediate threat to data security and require urgent remediation.
- **High:** Serious weaknesses that could be exploited but may require more effort or specific conditions.
- **Medium:** Moderate risks that should be addressed but are not immediately exploitable.
- **Low:** Minor issues or outdated practices that may not pose immediate risk but still warrant review.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a specific library type.

Algorithm Usage



The **Algorithm Usage** widget provides a comprehensive view of the cryptographic algorithms detected across your organization. It helps you understand the types of algorithms in use and their distribution across different components of your environment.

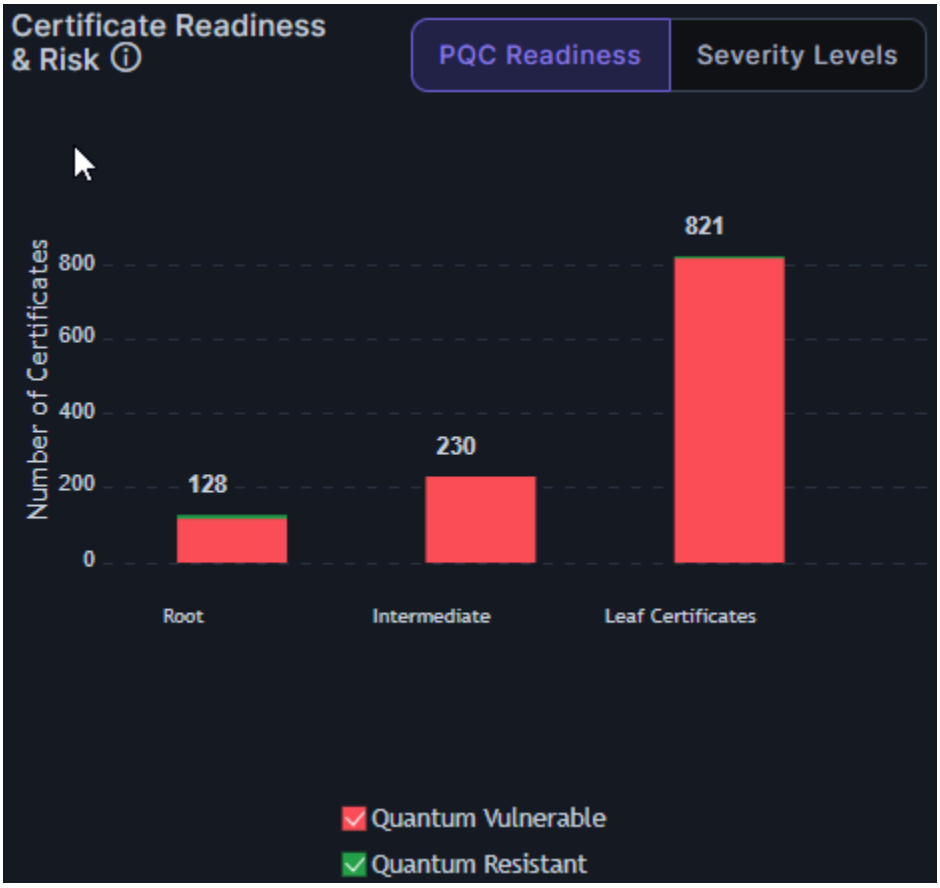
This donut chart visualization helps to identify the prevalence of specific algorithms such as RSA, AES, ECC, SHA variants, etc. and evaluate their suitability from a security and quantum-readiness perspective.

Use the dropdown menu in the top-right corner of the widget to update the visualization for a specific crypto asset, from the following options:

Option	Description
All	Displays the overall usage distribution of all cryptographic algorithms across all asset types
Certificates	Filters and shows only the algorithms used in digital certificates (e.g., signature and key exchange algorithms)
Configurations	Displays algorithms defined in application or system configurations (e.g., TLS settings, cipher suites)

Option	Description
Code	Shows algorithms found in source code, including hardcoded cryptographic logic or library references

Certificate Readiness & Risk



The **Certificate Readiness & Risk** widget provides a bar chart visualization of the certificates discovered across your organization, categorized either by their quantum readiness or by the severity of cryptographic risks they pose. This widget helps prioritize certificate upgrades and guides the transition toward quantum-safe cryptographic standards.

The stacked bar chart widget shows the following two views:

Option	Description
PQC Readiness	Displays certificates based on their readiness for post-quantum cryptography. Categories may include quantum resistant, partially ready, or quantum vulnerable certificates, depending on algorithm strength and key size.

Option	Description
Severity Levels	Shows certificates categorized by the severity of their cryptographic weaknesses critical, high, medium, or low based on outdated algorithms, key lengths, or insecure hashing/ signature schemes.

In each view, the chart plots the number of certificates corresponding to a PQC readiness level/severity level.

Use the tabs in the top-right corner of the chart to switch between the views.

List of Scans

The **List of Scans** section in the Quantum Trust Hub dashboards is, as the name suggests, a holistic list of all the PQC-focused scans run on your cryptographic environment.

Prerequisite: Verify that your user role has the required **ACF permission** to view list of scans. To enable ACF permission, click here.

For each scan type, this inventory lists the following details:

- Scan name
- Scan host IP
- Host FQDN
- Scan type
- User who triggered the scan
- Scan date
- Start time and end time of scan
- Crypto categories assessed as part of the scan
- Scan status

Common inventory functions

Feature	Description
Filters	To filter the inventory for viewing specific data: 1. From one or more of the following dropdown lists, select the required filtering criteria: • **Scan Type** • **Crypto Category** • **Status** 2. Click **Apply**.
Search	Enter free text or keywords to search for specific entries in the inventory.
Export	To export the inventory data: 1. Select at least one record from the inventory to export the corresponding data. 2. From the menu bar, click **Export**. 3. From the **How would you like to download the data?** Dialog box, select your preferred export file format (**CSV** or **XLS**). 4. Click **Submit**. The inventory data is downloaded to your local system as a zipped file.
Pagination	Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory.
Pagination Navigation	Use the pagination navigation buttons to move between the pages in the inventory.
Refresh	Use the **Refresh** button to reload the inventory to display the up-to-date records.

Code Scan

A PQC-focused code scan assess the quantum-readiness of your code. The **Code Scan** dashboard offers a consolidated view of quantum vulnerabilities in your codebase.

Prerequisite: Verify that your user role has the required **ACF permission** to view code scan reports. To enable ACF permission, click here.

The **Code Scan** dashboard, its projections, and the inventory generated are explained in detail here.

Configuration Scan

A PQC-focused configuration scan analyzes the quantum-readiness of the cryptographic settings configured across systems, applications, and the network infrastructure of your organization.

The **Configuration Scan** dashboard offers a holistic view of the quantum-readiness status and related vulnerabilities of your cryptographic settings.

Prerequisite: Verify that your user role has the required **ACF permission** to view configuration scan reports. To enable ACF permission, click here.

The **Configuration Scan** dashboard, its projections, and the inventory generated are explained in detail here.

Certificate Scan

A certificate scan assesses the PQC-readiness of the certificates in your cryptographic environment.

The **Certificate Scan** dashboard displays the quantum-readiness status and related vulnerabilities for your organization's digital certificates.

Prerequisite: Verify that your user role has the required **ACF permission** to view certificate scan reports. To enable ACF permission, click here.

The **Certificate Scan** dashboard, its projections, and the inventory generated are explained in detail here.

Quantum Trust Hub Inventory

The Quantum Trust Hub inventory offers consolidated details of the outcome of all three PQC-focused scan types. For each scan type, the inventory lists all cryptographic assets scanned, the corresponding quantum-readiness status of each, along with other details that are relevant to planning and prioritizing remediation activities for transitioning to quantum-safe alternatives.

Prerequisite: Verify that your user role has the required **ACF permission** to view inventory. To enable ACF permission, click here.

For detailed documentation of the **Code Scan** inventory, see Code Scan Inventory.

For the complete code scan documentation, see Quantum Trust Hub: Code Scan.

For detailed documentation of the **Configuration Scan** inventory, see Configuration Scan Inventory.

For the complete configuration scan documentation, see Quantum Trust Hub: Configuration Scan.

For detailed documentation of the **Certificate Scan** inventory, see Certificate Scan Inventory.

For the complete certificate scan documentation, see Quantum Trust Hub: Certificate Scan.

Quantum Trust Hub Policy

The **Policy** module in the Quantum Trust allows you to create, manage, and enforce policies to customize PQC scans according to your organization's requirements.

Prerequisite: Verify that your user role has the required **ACF permission** to view policies. To enable ACF permission, click here.

Note: The custom policies do not override the NIST standards, only the implementation of a scan for your organization.

For detailed documentation on custom PQC policies, see Configuring PQC Readiness/Post-Quantum Policies.

Chapter 7: Integrating and Operating the PQC Assessment Tool

Chapter 8: Scanning Cryptographic Assets for PQC Readiness

- Quantum Trust Hub: Code Scan
- Quantum Trust Hub: Certificate Scan
- Quantum Trust Hub: Configuration Scan

Quantum Trust Hub: Code Scan

A PQC-focused code scan assesses source code to detect the presence of quantum vulnerable cryptography embedded in code.

While certificates are as vulnerable to quantum attacks, they are easier to replace once their quantum-readiness has been assessed due to expiry. Code, however, lasts years, especially for embedded systems, financial applications, IoT, and firmware. Without code scanning, organizations risk leaving quantum vulnerable crypto buried in applications long after TLS certificates have been upgraded.

A PQC-focused code scan typically checks the quantum-readiness of the following elements in your code:

- Cryptographic methods
- Crypto libraries
- Crypto algorithms used in source code

Prerequisite: Verify that your user role has the required **ACF permission** to view code scan reports. To enable ACF permission, click here.

The **Code Scan** dashboard offers a consolidated view of your organization’s software code security posture, highlighting which parts of your codebase are quantum vulnerable.

To view the Code Scan dashboard, go to **Menu > Quantum Trust Hub > Dashboard > Code**.

The dashboard widgets are explained in the subsequent sections.

Quantum Readiness Score

The **Quantum Readiness Score** widget displays the cumulative PQC score for a code scan, aggregated from the PQC scores of all lines of code in your codebase. It indicates **how well the scanned code aligns with quantum-safe practices and policies**.

- Each line of code scanned is assigned a PQC score based on its quantum readiness.
- Quantum Resistant Crypto Categories Identified = **1** point
- Quantum Vulnerable Crypto Categories Identified = **0** points

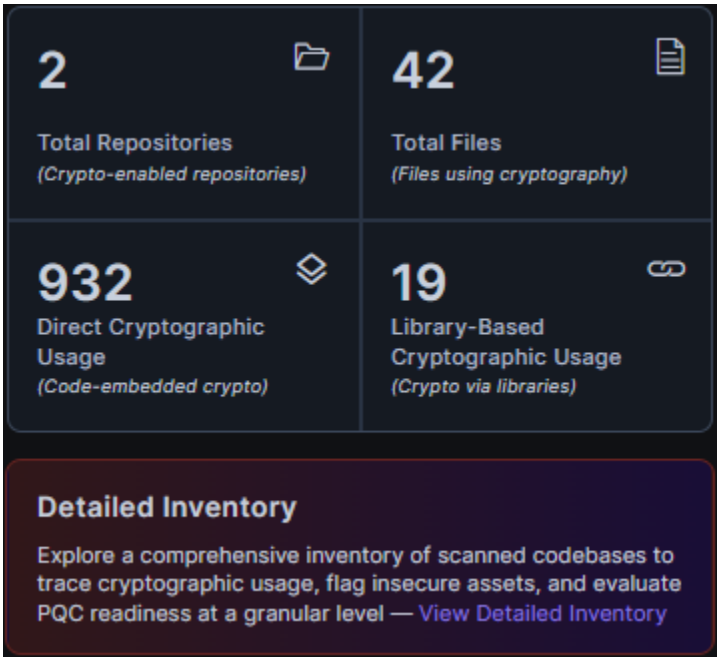
The code scan quantum readiness score, displayed using this widget, is then calculated as:

```
(Total Quantum Resistant Crypto Categories identified / Total Crypto Categories Identified) * 10
```

The threat level interpretation is therefore categorized as:

The threat level is displayed on the widget below the Gauge chart.
To read more on what the displayed threat level means and the recommended next steps, click **Know more** from the widget.

Code Scan Count Cards



The count cards on the Code Scan dashboard are used to display the following key metric values derived from the source code scan for PQC-readiness:

- **Total Repositories Scanned:** Total number of source code repositories scanned for PQC-readiness.

  A repository here refers to containers for source code, for example, GitHub, GitLab.

- **Total Files Scanned:** Total number of scanned files with crypto components
- **Total Cryptographic Usage:** Total number of instances of direct usage of cryptographic functions in the source code scanned across repositories.

  This count card is interactive; to view details of these instances, click the count card and you will be redirected to the **Direct Cryptographic Usage** tab in the **Code Scan** inventory.

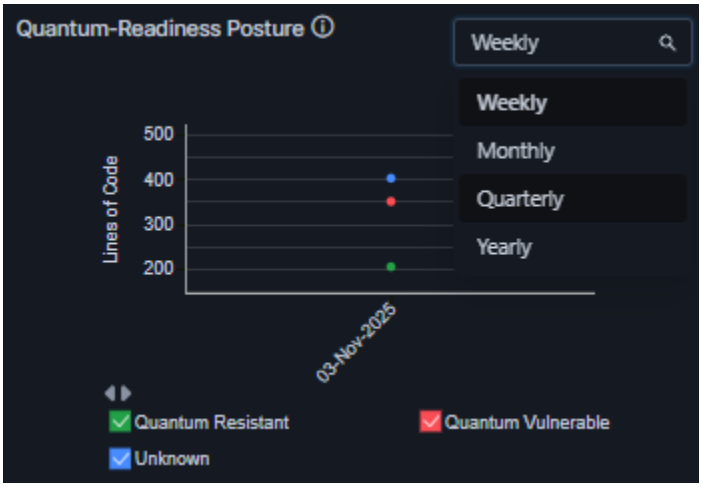
- **Total Cryptographic Dependencies:** Total number of cryptographic dependencies detected in the source code scanned across repositories.

  A cryptographic dependency refers to external libraries, modules, or packages that the source code relies on for performing cryptographic operations like encryption, decryption, key generation, and so on.

  The code scan checks these external cryptographic dependencies to determine how they impact the quantum-readiness of your application code.

  This count card is interactive; to view details of these instances, click the count card and you will be redirected to the **Cryptographic Dependencies** tab in the **Code Scan** inventory.

Quantum Readiness Posture

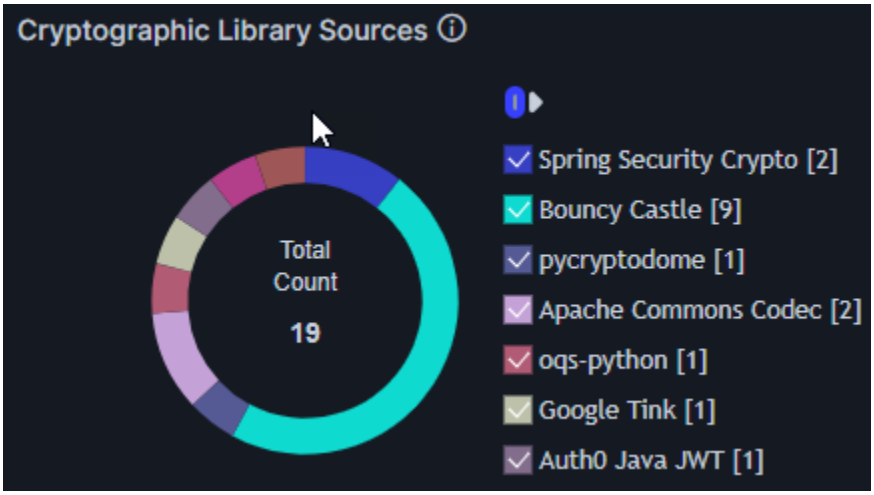


Based on the results of the PQC-focused source code scan, the **Quantum Readiness Posture** chart indicates the PQC-readiness of your codebase for resisting potential quantum threats.

The chart plots the number of crypto assets scanned for a selected duration, and represents the code that is quantum safe (with a green line) and quantum vulnerable (with a red line), as well as code whose quantum safety status could not be determined (with a blue line).

You can select the duration to be plotted on the Y-axis from the dropdown list in the top-right corner of the widget. The chart legends are interactive. Select/clear the checkbox for a quantum safety status value to show/hide, respectively, the corresponding data on the chart.

Cryptographic Library Sources

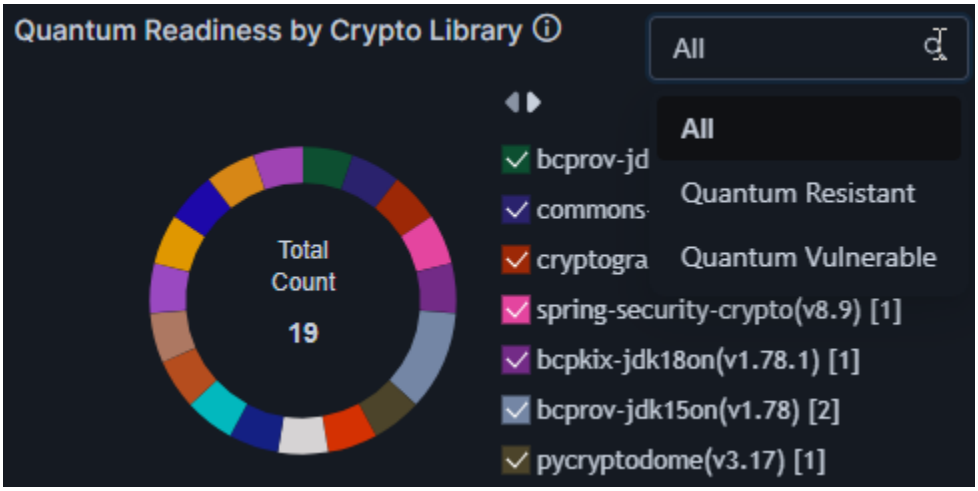


The **Cryptographic Library Sources** donut chart shows the distribution of external cryptographic libraries usage across repositories. The chart plots the data that is displayed in the **Cryptographic Dependencies** tab in the **Code Scan** inventory.

To view the data specific to a cryptographic library, click the corresponding donut slice. You will be redirected to the **Cryptographic Dependencies** tab in the **Code Scan** inventory filtered for the selected library.

The legend lists the cryptographic libraries detected and the usage count for each. Use the interactive legend to filter the visualization for specific libraries.

Quantum Readiness by Crypto Library



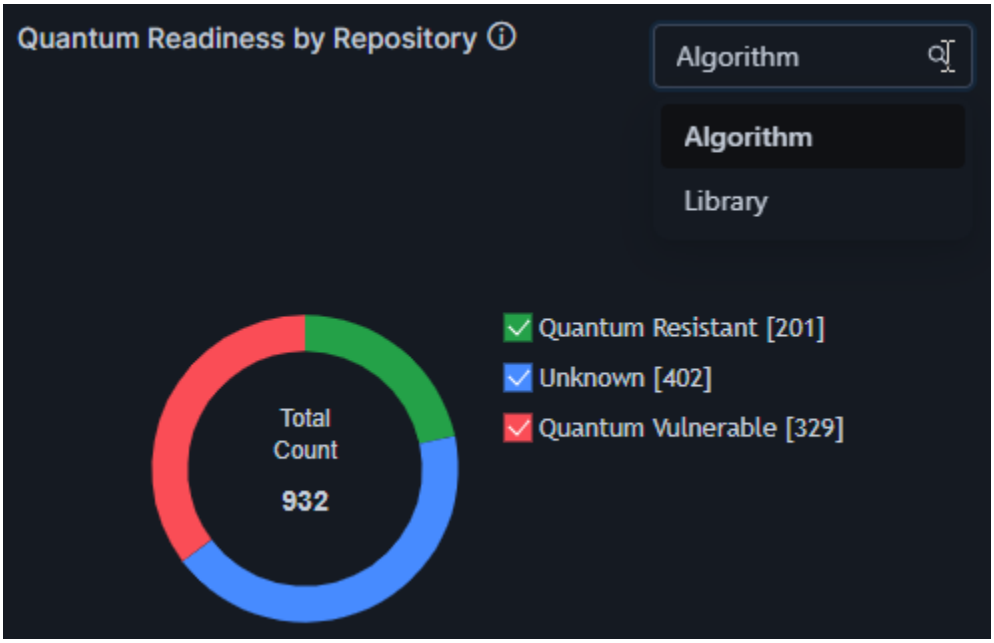
The **Quantum Readiness by Crypto Library** chart shows the quantum readiness of each cryptographic library in your environment.

Use the dropdown list in the top-right corner of the chart to filter the chart data, showing only quantum resistant libraries, quantum vulnerable libraries, or both.

The chart elements are interactive. To view the repositories associated with each cryptographic library, click the corresponding donut slice. You will be redirected to the **Cryptographic Dependencies** tab in the **Code Scan** inventory filtered for the selected library.

The chart legends are also interactive. Select/clear the checkbox for a cryptographic library to show/hide, respectively, the corresponding data on the chart.

Quantum Readiness by Repository

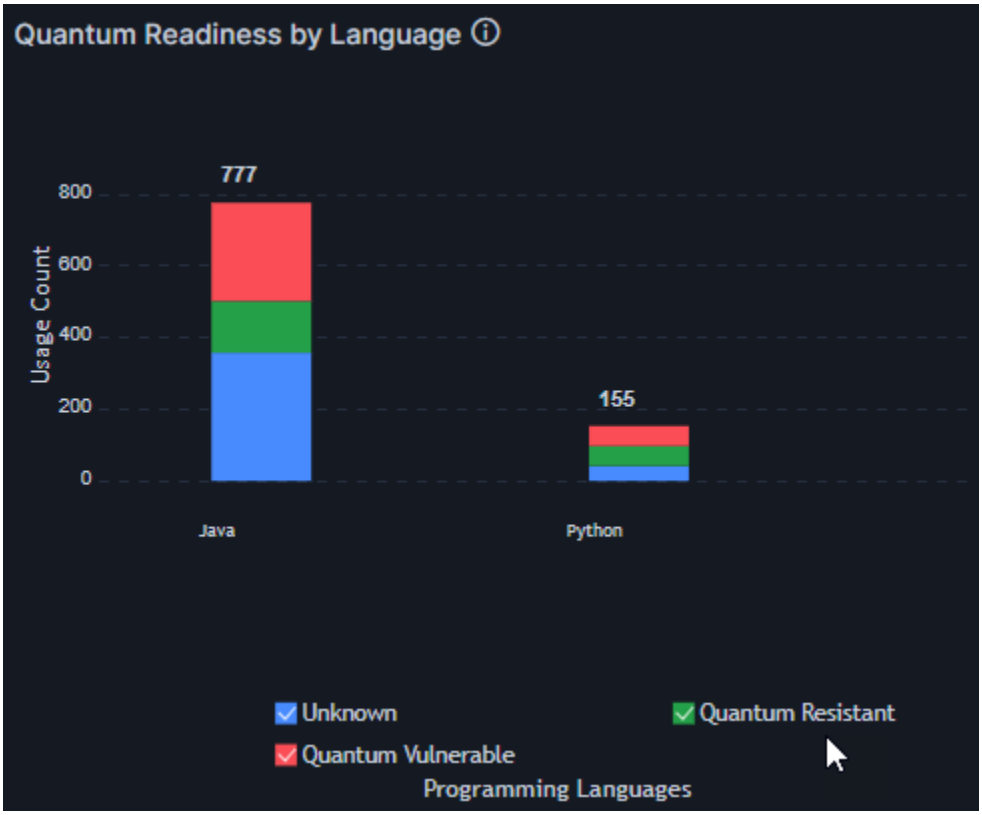


The **Quantum Readiness by Repository** donut chart shows the distribution of quantum vulnerable and quantum resistant algorithms and libraries in your environment. Use the dropdown list to filter the chart data for algorithms and libraries, as required.

The chart elements are interactive. To view the data for a quantum safety status, click the corresponding donut slice. For repository type **Algorithm**, you will be redirected to the **Direct Cryptographic Usage** tab in the **Code Scan** inventory, filtered for the selected quantum safety status. For repository type **Library**, you will be redirected to the **Cryptographic Dependencies** tab in the **Code Scan** inventory filtered for the selected quantum safety status.

The chart legends are also interactive. Select/clear the checkbox for a quantum safety status value to show/hide, respectively, the corresponding data on the chart.

Quantum Readiness by Language

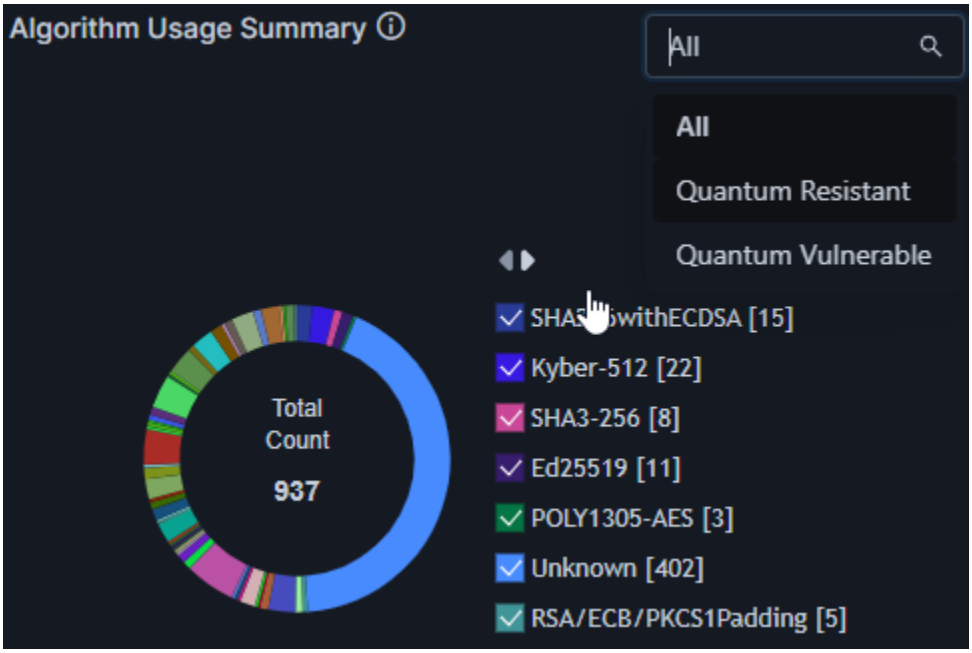


The **Quantum Readiness by Language** stacked bar chart shows quantum safety status according to the programming languages used in your codebase.

The chart elements are interactive. To view the data for the programming languages that correspond to a quantum safety status, click the associated bar in the chart. You will be redirected to the **Direct Cryptographic Usage** tab in the **Code Scan** inventory filtered for the selected quantum safety status.

The chart legends are also interactive. Select/clear the checkbox for a quantum safety status value to show/hide, respectively, the corresponding data on the chart.

Algorithm Usage Summary



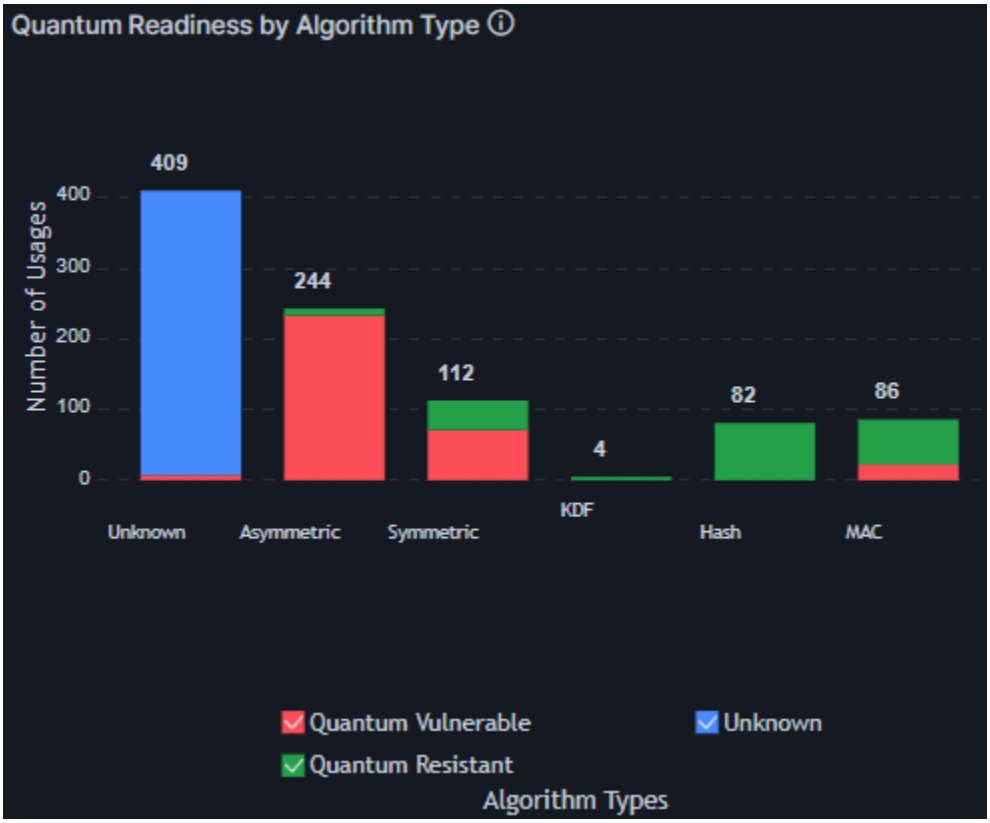
The **Algorithm Usage Summary** donut chart shows a snapshot of the different algorithms used in your codebase.

Use the dropdown list in the top-right corner of the chart to filter the data, showing only quantum resistant algorithms, quantum vulnerable algorithms, or both.

The chart elements are interactive. To view the data for an algorithm, click the associated slice in the donut chart. You will be redirected to the **Direct Cryptographic Usage** tab in the **Code Scan** inventory filtered for your selection.

The chart legends are also interactive. Select/clear the checkbox for an algorithm to show/hide, respectively, the corresponding data on the chart.

Quantum Readiness by Algorithm Type



The **Quantum Readiness by Algorithm Type** column chart shows quantum safety status according to the different algorithm types used in your codebase.

The chart elements are interactive. To view the data for algorithms that correspond to a quantum safety status, click the associated bar in the chart. You will be redirected to the **Direct Cryptographic Usage** tab in the **Code Scan** inventory filtered for the selected quantum safety status.

The chart legends are also interactive. Select/clear the checkbox for a quantum safety status value to show/hide, respectively, the corresponding data on the chart.

- Code Scan Inventory

Code Scan Inventory

Prerequisite: Verify that your user role has the required **ACF permission** to view code scan inventory. To enable ACF permission, click here.

Viewing the Code Scan Inventory

To view the code scan inventory:

To view the Code Scan inventory, go to **Menu > Quantum Trust Hub > Inventory > Code**.
You will be redirected to the **Code Scan Inventory**.

The code scan inventory is divided into three tabs:

- **Direct Cryptographic Usage**

  This tab displays the scan results for cryptographic functions that are called directly from your application's first-party code. For details, see Direct Cryptographic Usage.

- **Cryptographic Dependencies**

  This tab displays the scan results for the external libraries, modules, or packages that the source code relies on for performing cryptographic operations. For details, see Cryptographic Dependencies.

- **Upload Custom Library**

  This tab lets you upload a custom defined class or method, which is not a direct call of the supported libraries, for PQC assessment scanning. For details, see Uploading Custom Libraries for Code Scanning.

Common Inventory Functions

The table below explains the inventory functions that are common to both tabs, **Direct Cryptographic Usage** and **Cryptographic Dependencies**.

Feature	Description
Filters	To filter the inventory for viewing specific data: 1. From one or more of the following dropdown lists, select the required filtering criteria: • **Quantum Readiness** • **Severity** • **Repository** 2. Click **Apply**.
Search	Enter free text or keywords to search for specific entries in the inventory.

Feature	Description
Export	To export the inventory data: 1. Select at least one record from the inventory to export the corresponding data. 2. From the menu bar, click **Export**. 3. From the **How would you like to download the data?** Dialog box, select your preferred export file format (**CSV** or **XLS**). 4. Click **Submit**. The inventory data is downloaded to your local system as a zipped file.
Pagination	Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory.
Pagination Navigation	Use the pagination navigation buttons to move between the pages in the inventory.
Refresh	Use the **Refresh** button to reload the inventory to display the up-to-date records.

Direct Cryptographic Usage

Direct cryptographic usage refers to the algorithms, class names, and methods mentioned in the source code.

The **Direct Cryptographic Usage** tab in the code scan inventory displays the following details for all code scanned across repositories:

Column descriptions for the Direct cryptographic usage inventory

Column	Description
Repo Name	Name of the repository where the scanned code is located.
File path	Location of the code file within the repository.
Class name	Class within the file where the cryptographic operation is implemented.
Method name	Method that invokes the cryptographic operation.
Language	Programming language used to write the code.
Line number	Line number in the code where the cryptographic call is written.

Column descriptions for the Direct cryptographic usage inventory (continued)

Column	Description
Crypto Category	Type of cryptographic item detected in the code.
Algorithm Name	[For **Crypto Category** = **Algorithm]** Algorithm invoked via the code
Algorithm Type	[For **Crypto Category** = **Algorithm]** Algorithm type (Asymmetric, Symmetric, Message Authentication Code, and so on) invoked via the code
Severity	Level of risk posed by the cryptographic operation scanned.
Quantum Readiness	Quantum readiness status of the crypto category detected.
Recommendation Action	Suggested next steps, according to the severity and the quantum readiness status.

Cryptographic Dependencies

Code scan for cryptographic dependencies refers to the assessment of external libraries, packages, and frameworks that your code depends on. Your code doesn't call the crypto functions directly; it inherits the cryptography from these external dependencies.

The **Cryptographic Dependencies** tab in the code scan inventory displays the following details for all code scanned across repositories:

Column descriptions for the cryptographic dependencies inventory

Column	Description
Repo Name	Name of the repository where the scanned code is located.
File path	Location of the code file within the repository.
Library name	Cryptographic library (external) used.
Version	Version number of the cryptographic library detected (required to assess the quantum safety of the library).
Crypto Category	Type of cryptographic item detected in the code.
Cryptographic Library Source	Source of the cryptography library.
Quantum Readiness	Quantum readiness status of the cryptographic category and library detected.

Column descriptions for the cryptographic dependencies inventory (continued)

Column	Description
Recommendation action	Suggested next steps, according to the severity and the quantum readiness status.

Uploading Custom Libraries for Code Scanning

In addition to scanning in-built as well as external cryptographic libraries, AppviewX lets you upload a custom defined class or method that is not a direct method call of the supported libraries for assessing its crypto compliance. For example, a wrapper class that is built on top of Bouncy Castle or a new library defined by a user that is not currently supported by AppViewX.

To do this:

1. Go to **Menu > Quantum Trust Hub > Inventory > Code**.
   The **Code Scan Inventory** page is displayed.
2. From the menu bar, go to the **Upload Custom Library** tab.
   The **Get Started with Custom Library Upload** page is displayed.
3. Click **Download Sample Template** and save the CSV template file on your local machine.
4. In the CSV template file, enter the following details as relevant to your custom library:
   - Class name
   - Method name
   - Language
   - Library name
   - Algorithm used

   Note: Class name and **Method name** are mandatory inputs for the Java and Python languages. The class name of Python is expected to be a fully classified name from the package. For example **crypto.example.aes**. Class name can be skipped for C and CPP.

5. On the **Get Started with Custom Library Upload** page, to upload the filled template file from your local machine:

   - **Drag & Drop** the template file from your local machine.

     OR

   - Click  to upload the file from your local machine.

6. Test

   The **Custom Library Preview** is updated automatically based on the library details you uploaded.

Viewing the Custom Library Scan Results

The table below explains the inventory functions in the **Custom Library Preview**.

Feature	Description
Search custom library data	Enter free text or keywords to search for specific entries in the inventory.
Pagination	Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory.
Pagination Navigation	Use the pagination navigation buttons to move between the pages in the inventory.
Refresh	Use the **Refresh** button to reload the inventory to display the up-to-date records.

For the custom library scanned, the **Custom Library Preview** displays the following details:

Column descriptions for the custom libraries

Feature	Description
Class Name	Class in the custom library that contains the cryptographic logic.
Method Name	Method in which the cryptographic logic is used.
Language	Programming language in which the custom library is written.
Library Name	Name of the custom library.
Algorithm	Cryptographic algorithm used for the operation.
Quantum Readiness	Quantum safety status of the custom library.
Recommendations	Suggested next steps, according to the quantum readiness status.

Quantum Trust Hub: Certificate Scan

Certificate Scan: Overview

A certificate scan is the process of assessing an organization's cryptographic environment to find all digital certificates currently in use. In the context of a post-quantum cryptography (PQC) implementation, a certificate scan helps to build a baseline inventory that can be used to evaluate an PQC-readiness of an organization's certificates.

Results of the PQC certificate scan can be used for identifying, classifying, and scoring certificates based on their quantum resistance, enabling informed decision-making and prioritization of remediation efforts.

For every certificate discovered via a certificate scan, the following parameters are evaluated:

- Public key algorithm used for encryption, digital signatures, or key exchange (for example, RSA, ECC, PQC)
- Cryptographic key size (for example, RSA-1024, RSA-2048, RSA-4096, etc.)
- Hash function (for example, MD5, SHA-1, SHA-2, SHA-3)

Benefits of the PQC certificate scan

- A PQC-focused certificate scan helps to identify weak algorithms, smaller key sizes, and insecure hashes that can be exploited by hackers.
- Based on the above classification, your organization can identify and prioritize high-risk certificates for immediate action.

Certificate Classification and PQC Risk Severity Assignment

The outcome of this analysis helps classify certificates as:

- **Classical** (certificates using only the traditional public-key algorithms like RSA/ECC)
- **Hybrid** (certificates that use classical algorithms along with the PQC algorithms)
- **PQC-only** (certificates that use only PQC algorithms)

Based on the findings of this evaluation, the PQC risk severity is assigned as follows:

PQC Risk Severity	Criteria
Critical	• RSA < 2048-bit • ECC < 224-bit

PQC Risk Severity	Criteria
	• DSA (deprecated) • Hashes: MD5, SHA-1
High	• RSA-2048, ECC-P256 (secure now, but not quantum safe) • Hashes: SHA-256 / SHA-384 (secure today, but classical)
Medium	• RSA-3072, ECC-P384 (stronger, but still classical) • PQC-Hybrid (RSA/ECC + PQC algorithm)
Low (Quantum Safe)	• PQC-only certificates (e.g., Kyber, Dilithium once standardized)

The outcome of the PQC certificate scan is represented using two mediums:

- A dashboard that uses multiple widgets to present the data across different dimensions.
- A certificate scan inventory that lists all the certificates discovered in a scan along with their PQC risk severity and quantum readiness.

Certificate Scan Dashboard

The **Certificate Scan** dashboard is a collection of widgets that display a multi-faceted view of the PQC risk and quantum readiness data for the results of a PQC-focused certificate scan. The data displayed on the dashboard is a quantifiable measure of your organization's risk and readiness of the PQC adoption.

Prerequisite: Verify that your user role has the required **ACF permission** to view certificate scan reports. To enable ACF permission, click here.

To view the Certificate Scan dashboard, go to **Menu > Quantum Trust Hub > Dashboard > Certificate**.

The dashboard widgets are explained in the subsequent sections.

Quantum Readiness Score

The **Quantum Readiness Score** widget displays the cumulative PQC score for a scan, aggregated from the PQC scores of all discovered certificates.

- Each discovered certificate is assigned a PQC score based on its quantum resistance.
- Quantum resistant certificate = **1** point
- Hybrid certificate = **0.6** points
- Quantum vulnerable certificate = **0** points

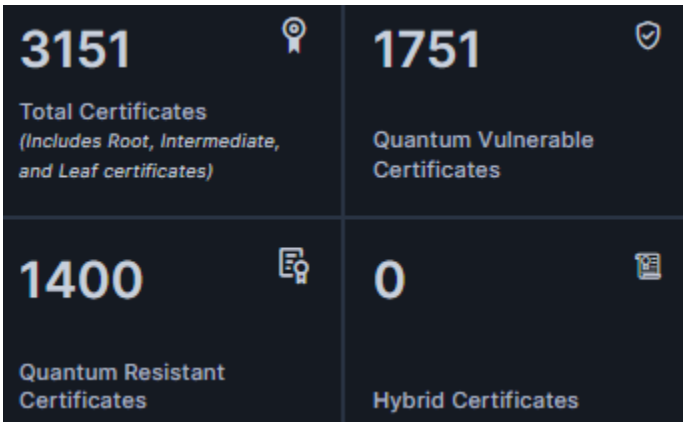
The **Certificate PQC score**, displayed using this widget, is then calculated as:

```
(Sum of all certificate points/Total number of certificates discovered in the scan) * 10
```

The threat level interpretation is therefore categorized as:

The threat level is displayed below the Gauge chart.
To read more on what the displayed threat level means and the recommended next steps, click **Know more** from the widget.

Certificate Count



The certificate count metric card displays the total number of certificates scanned for quantum readiness and, therefore, categorized as

- Quantum vulnerable (certificates that rely purely on classical algorithms like RSA and ECC)
- Quantum resistant (certificates use Post-Quantum Cryptographic (PQC) algorithms)
- Hybrid (certificates that combine a classical algorithm with a PQC algorithm)

To view the details of the certificates under each of these categories, click the corresponding block on the metric card.

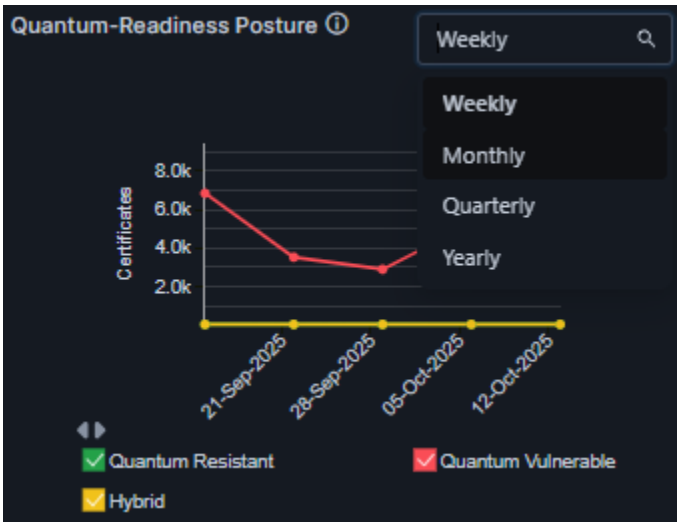
A **Certificate Inventory** pop-up, filtered for the selected certificate type block, is displayed, with the complete details of each certificate of that type.

For example, to view all quantum vulnerable certificates, click the **Quantum Vulnerable Certificates** block from the metric card. The corresponding **Certificate Inventory** is displayed.

In this certificate inventory, you can:

- View detailed certificate data, for the selected certificate type, organized by certificate category (server, client, code signing, root, intermediate, and device).
- Switch between certificate categories by clicking the corresponding tab from the menu bar.
- Export the certificate inventory from the Quantum Trust Hub by selecting the records you want to export, clicking **Export** and finally, selecting the column display and format of the exported certificate data.

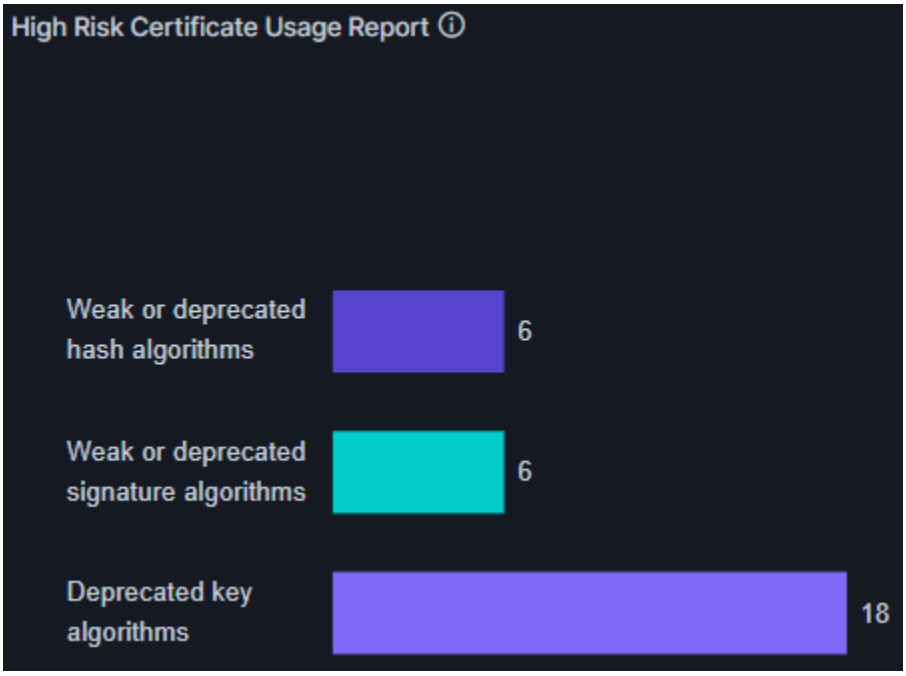
Quantum-Readiness Posture



The **Quantum-Readiness Posture** chart displays a overview of your organization's readiness for a PQC transition for a selected period of duration. Select the duration from a dropdown list in the top-right corner of the widget.

The chart legends are interactive. Select/clear the checkbox for a risk severity level to show/hide, respectively, the corresponding data on the chart.

High Risk Certificate Usage Report



This chart highlights cryptography algorithm weakness/deprecations across the quantum vulnerable certificates in your system. This helps to identify areas of security risk associated with weak, deprecated, or outdated algorithms.

The info icon displayed next to the widget title gives a breakdown of the hashes, signatures, and key exchange details for the algorithms detected by the scan.

Each bar in this chart is interactive. To view details of the certificates that subscribe to the corresponding cryptographic algorithm weakness/deprecation, click the corresponding bar on the chart. In the Certificate Inventory, you can view:

- detailed certificate data, for the selected certificate type, organized by certificate category (server, client, code signing, root, intermediate, and device).
- switch between certificate categories by clicking the corresponding tab from the menu bar.

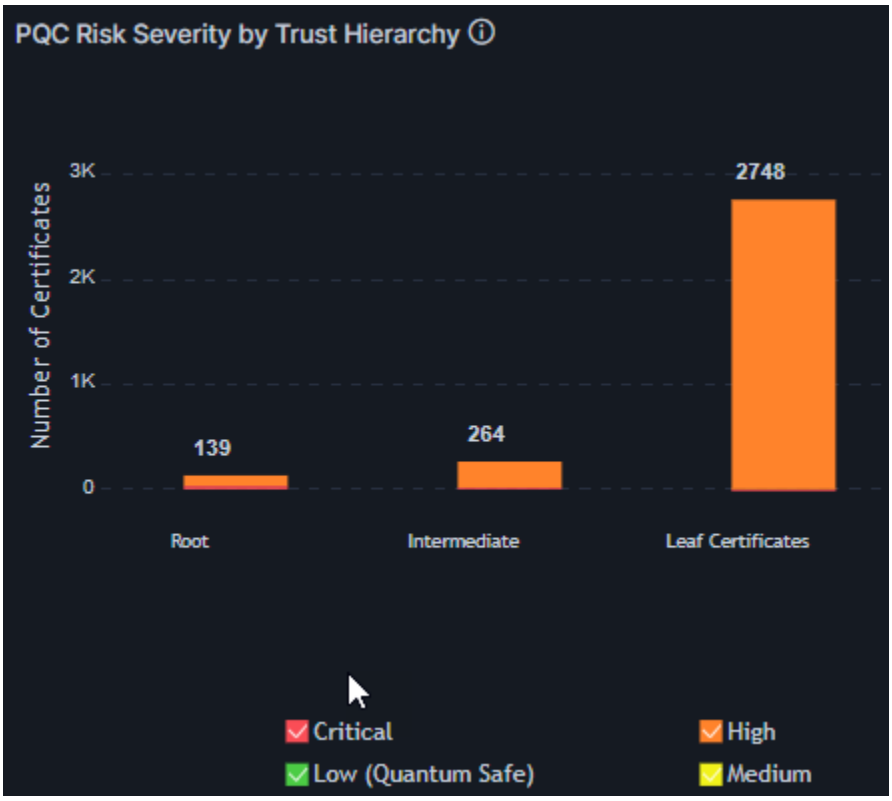
A **Certificate Inventory** pop-up, filtered for your selection, is displayed.

In this certificate inventory, you can:

- View detailed certificate data, for the selected certificate type, organized by certificate category (server, client, code signing, root, intermediate, and device).
- Switch between certificate categories by clicking the corresponding tab from the menu bar.
- Export the certificate inventory from the Quantum Trust Hub.

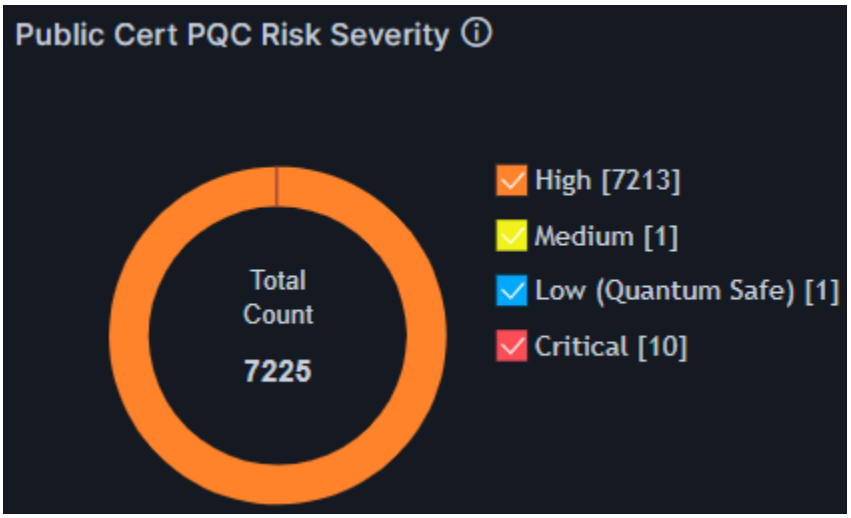
  To do this:
  1. Select the checkboxes corresponding to the records you want to export.
  2. From the toolbar, click **Export**.
  3. From the **How would you like to download the data?** dialog box, select the column display and the file format for the exported configuration data.
  4. Click **Submit**.

PQC Risk Severity by Trust Hierarchy



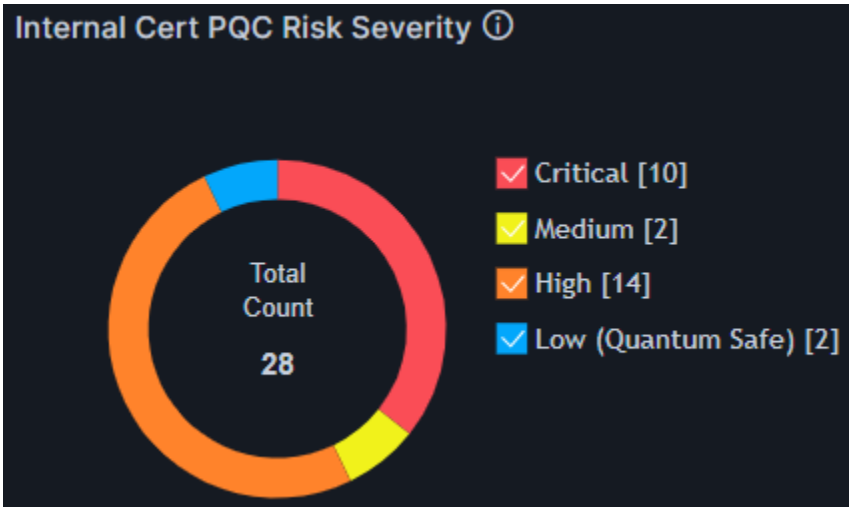
This chart displays the distribution of the PQC risk severity across the certificate hierarchy (root, intermediate, and leaf certificates), for the quantum vulnerable certificates. The X- axis indicates risk severity and Y axis indicates the number of certificates.You can filter the data in the chart by risk severity level.

Public Cert PQC Risk Severity



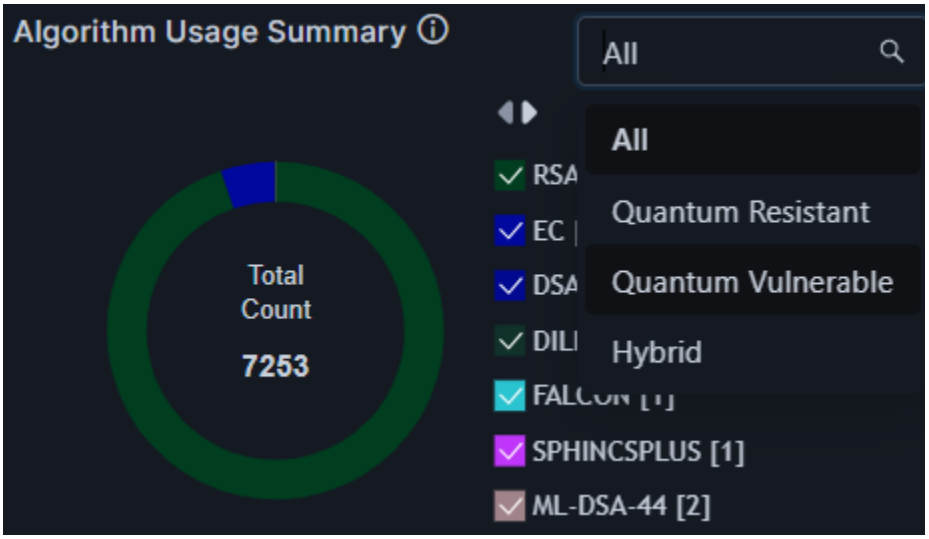
This is a distinct visualization of the risk readiness of public certificates. This widget displays the risk severity breakdown for the total number of public certificates scanned. The chart legends are interactive. Select/clear the checkbox for a risk severity level to show/hide, respectively, the corresponding data on the chart.

Internal Cert PQC Risk Severity



This widget displays the risk severity breakdown for the total number of private certificates scanned. You can filter the data in the chart by risk severity level.

Algorithm Usage Summary

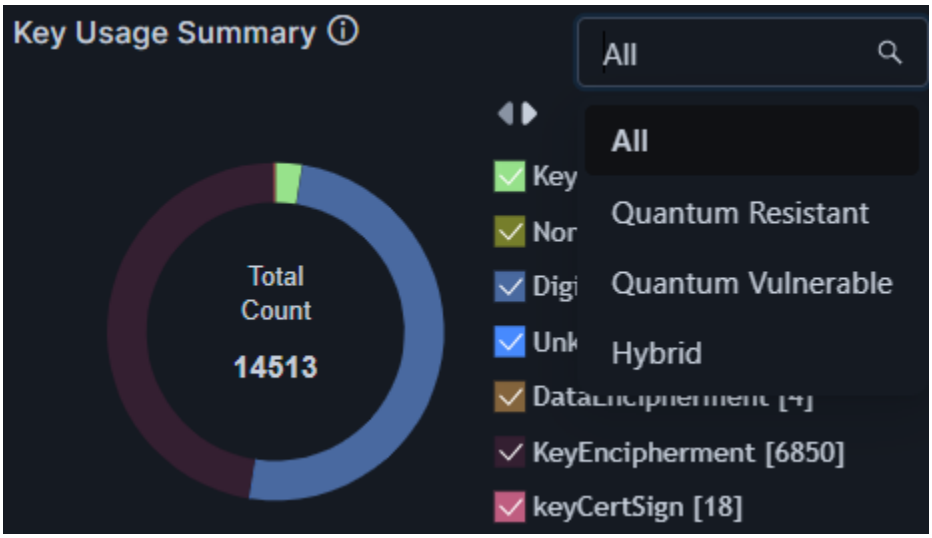


This widget displays a breakdown of the cryptographic algorithms used across the certificates scanned.

The interactive chart legend displays the full list of algorithms used along with the count of instances. Use the legend interactivity to filter the visualization for specific algorithms.

Use the dropdown list from the top-right corner of the widget to filter the chart visualization for a quantum safety status value (**All**, **Quantum Resistant**, **Quantum Vulnerable**, **Hybrid**).

Key Usage Summary

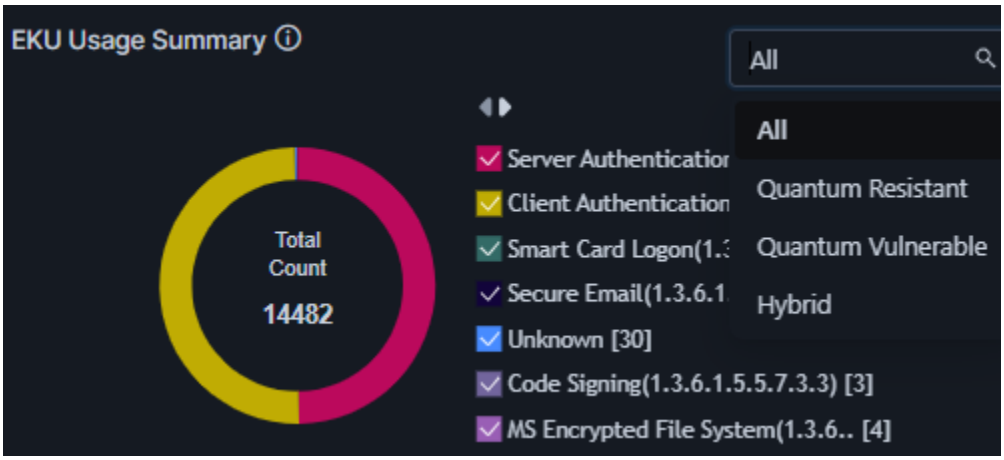


This widget displays the distribution of how cryptographic keys are used across the certificates scanned.

The interactive chart legend displays the full list of the keys used along with the count of instances. Use the legend interactivity to filter the visualization for specific key types.

Use the dropdown list from the top-right corner of the widget to filter the key exchange visualization according to the quantum-safety status (**All**, **Quantum Resistant**, **Quantum Vulnerable**, **Hybrid**).

EKU Usage Summary



This widget shows a breakdown of the specific application purposes for which the scanned certificates are valid. This is an important insight that indicates risk severity with respect to business functions.

The interactive chart legend displays the full list of certificate EKU usages, used along with the count of instances. Use the legend interactivity to filter the visualization for specific certificate usages.

For a filtered view, use the dropdown list from the top-right corner of the widget to select a PQC-certificate category (**All**, **Quantum Resistant**, **Quantum Vulnerable**, **Hybrid**).

• Certificate Scan Inventory

Certificate Scan Inventory

Prerequisite: Verify that your user role has the required **ACF permission** to view code scan inventory. To enable ACF permission, click here.

Viewing the Certificate Scan Inventory

To view the certificate scan inventory:

To view the certificate scan inventory, go to **Menu > Quantum Trust Hub > Inventory > Certificate**. You will be redirected to the **Certificate Scan Inventory** in the Quantum Trust Hub.

This certificate inventory provides a category-wise comprehensive view of all certificates within your organization's cryptographic infrastructure. For a PQC-focused analysis, you can customize the inventory view to display only the columns that are relevant to your requirement.

AppViewX has introduced the following new columns to indicate the PQC readiness of your certificates and to enable prompt remediation, as required:

Column descriptions for the certificate scan inventory

Column	Description
PQC Risk Severity	**PQC Risk Severity** indicates how severely a certificate could be impacted by a potential quantum attack. Based on the certificate PQC score calculated for a certificate, this column displays one of the following values: • **Critical** • **High** • **Medium** • **Low (Quantum Safe)**
Quantum Readiness	Quantum readiness is a measure of an organization's preparedness to address the impact of quantum computing over cryptography. Your certificate infrastructure will be considered quantum ready when it is deemed capable of protecting your data, systems, and communication against the threats posed by quantum computers to today's encryption methods. This column in the inventory indicates the quantum readiness of individual certificates using the following values: • **Quantum Resistant** (certificate uses PQC algorithms for encryption that can withstand attacks from classical as well as quantum computers) • **Quantum Vulnerable** (certificate uses classical encryption algorithms that can be broken by quantum computers) • **Hybrid** (certificate uses a cryptographic approach that combines the current classical algorithms with PQC algorithms)

For instructions on customizing the certificate inventory view, click here.

Customizing Columns in the Server Certificate Inventory

This topic will be listed here (please scroll to the end of this page) for the server certificate inventory. Similarly, this same topic will be added for the client, code signing, root, intermediate, and device certificate inventory documentation.

For the certificate inventory, AppViewX lets you show/hide columns in the certificate inventory based on the data you want to display. Selecting which columns to display lets users focus on relevant data, improves readability, and speeds up analysis and reporting.

To customize the columns in the server certificate inventory:

1. Go to **Menu** > **CERT+** > **Certificate Inventory** > **Server**.
   The **Server Certificate** inventory is displayed.
2. From the toolbar, click **Columns**.
   The **Columns** dialog box is displayed.
3. Select/Clear the checkboxes corresponding to the columns you want to show/hide in the inventory.

> **Tip:**
> • You can also use the **Search** field to search for the required columns.
> • To select all columns, select the **Select** all checkbox from the dialog box.
> • To reset your changes, click **Reset to previous column selection**.

> **Note:** Columns marked with an * are mandatory for displaying in the inventory and hence cannot be hidden.

4. Click **Save**.
   The inventory is updated according to your selection.

Customizing Columns to View PQC Readiness

Starting v2025.0.0.0, AppViewX introduces PQC capabilities for CERT+ that will review and provide a holistic view of the PQC readiness of your certificate inventory.

PQC Readiness is a measure of your organization's preparedness for transitioning to PQC algorithms to resist probable attacks from quantum computers.

To be able to view the PQC readiness of your certificates, you will be required to enable the following columns for display in the certificate inventory:

Column	Description
PQC Risk Severity	**PQC Risk Severity** indicates how severely a certificate could be impacted by a potential quantum attack. Based on the certificate PQC score calculated for a certificate, this column displays one of the following values: • **Critical** • **High** • **Medium** • **Low (Quantum Safe)**
Quantum Readiness	Quantum readiness is a measure of an organization's preparedness to address the impact of quantum computing over cryptography. Your certificate infrastructure will be considered quantum ready when it is deemed capable of protecting your data, systems, and communication against the threats posed by quantum computers to today's encryption methods. This column in the inventory indicates the quantum readiness of individual certificates using the following values: • **Quantum Resistant** (certificate uses PQC algorithms for encryption that can withstand attacks from classical as well as quantum computers) • **Quantum Vulnerable** (certificate uses classical encryption algorithms that can be broken by quantum computers) • **Hybrid** (certificate uses a cryptographic approach that combines the current classical algorithms with PQC algorithms)

Quantum Trust Hub: Configuration Scan

A **Configuration Scan** is the process of analyzing the cryptographic settings and parameters configured across systems, applications, and network infrastructure within an organization. In the context of **post-quantum cryptography (PQC)** implementation, a configuration scan helps build a comprehensive view of cryptographic configurations that may be vulnerable to quantum-based attacks.

The goal of a PQC configuration scan is to identify and assess systems that are using classical algorithms (such as RSA or ECC), outdated protocol versions, or insecure cipher suites, which may not provide adequate protection in a post-quantum world.

The results of the scan support identifying, classifying, and scoring systems based on their cryptographic posture and PQ-readiness. This enables informed planning and prioritization of remediation activities, such as protocol upgrades, algorithm replacements, or configuration hardening.

For each system or application analyzed in a configuration scan, the following parameters are typically evaluated:

- **Enabled cryptographic algorithms and protocols** (for example, RSA, ECC, TLS 1.2, TLS 1.3, IPsec, SSH)
- **Cipher suites** (e.g., inclusion of PQC-safe or hybrid suites, use of deprecated ciphers like RC4 or 3DES)
- **Key sizes and cryptographic parameters** (e.g., RSA-2048, ECDSA P-256)
- **Protocol versions and fallback behavior** (e.g., support for only secure versions like TLS 1.3)
- **Support for PQC or hybrid cryptography mechanisms**, where available.

Benefits of the PQC Configuration Scan

- Helps identify cryptographic configurations that rely on quantum vulnerable algorithms and protocols.
- Reveals misconfigurations or weak parameter settings that may increase security risks.
- Assists in prioritizing updates to cryptographic settings in alignment with **NIST guidelines** (e.g., NIST SP 800-131A, SP 800-52r2).
- Supports cryptographic agility and prepares systems for a seamless transition to PQC-safe algorithms.

The **Configuration Scan** dashboard is a collection of widgets that display a multi-faceted view of the PQC risk and quantum readiness data for the results of a PQC-focused configuration scan. The data displayed on the dashboard is a quantifiable measure of your organization’s risk and readiness for the PQC adoption.

Prerequisite: Verify that your user role has the required **ACF permission** to view configuration scan reports. To enable ACF permission, click here.

To view the **Configuration Scan** dashboard:

1. Go to **Menu > Quantum Trust Hub > Dashboard**.

   The **Quantum Trust Hub : Organization View** page is displayed.
2. From the menu bar, select **Configuration**.

   The **Quantum Trust Hub : Configuration Scan** page is displayed.

The dashboard widgets are explained in the subsequent sections.

Quantum Readiness Score

The **Configuration Scan Quantum Readiness Score** widget displays the cumulative PQC score for a scan, aggregated from the PQC scores of all configurations.

- Each **Configuration Scan** is assigned a PQC score based on its quantum resistance.
- Quantum Resistant Categories (1 point each) = 5
- Total Categories Assessed: 10
- Quantum Vulnerable = 0 Point

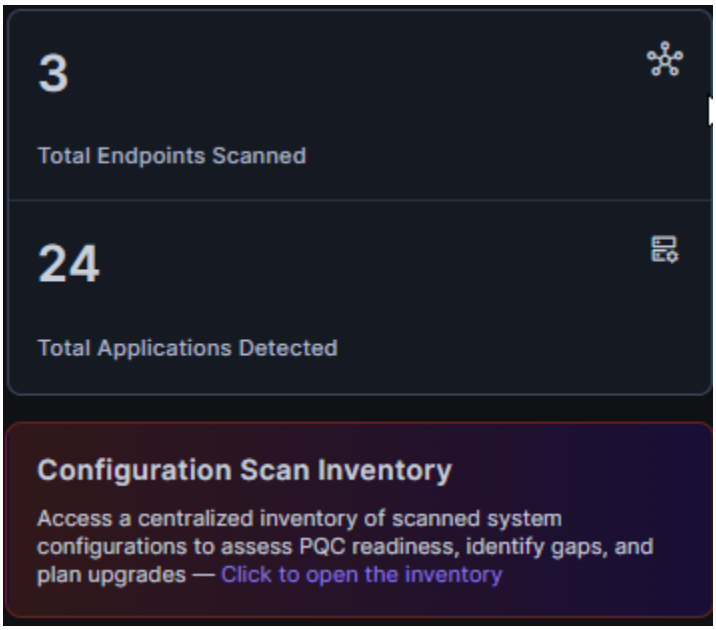
The **Configuration Scan Quantum Readiness score**, displayed using this widget, is then calculated as:

```
Your Total PQC Score = (Number of Quantum Resistant Crypto Categories Identified) /
(Total Number of Crypto Categories Identified)
```

The threat level interpretation is therefore categorized as:

The threat level is displayed on the widget below the Gauge chart.
To read more on what the displayed threat level means and the recommended next steps, click **Know more** from the widget.

Configuration Count



The Configuration Count widget provides a high-level summary of your environment's cryptographic configuration scan results. It helps users quickly understand the scope and coverage of scanned assets, enabling better assessment of cryptographic risk and quantum readiness.

Displayed Metrics

Metric	Description
Total Endpoints Scanned	Displays the total number of **individual endpoints** (e.g., servers, devices, network assets) that were scanned during the configuration assessment. This reflects the breadth of your cryptographic visibility across the infrastructure.
Total Applications Detected	Indicates the number of **distinct applications** discovered and analyzed across the scanned endpoints. These applications are using cryptographic configurations that were evaluated for vulnerabilities and quantum readiness.

To view the details list of the configuration scan under these categories, click the corresponding block on the metric card.

A **Configuration Inventory** pop-up, filtered for the selected configuration type block, is displayed, with the complete details of each configuration type.

For example, to view the Total Endpoints Scanned, click the **Total Endpoints Scanned** block from the metric card. The corresponding **Configuration Inventory** is displayed.

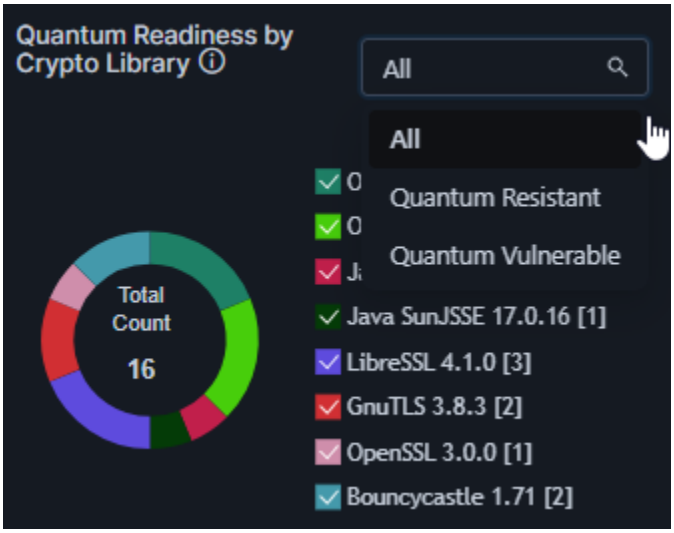
In this configuration inventory, you can:

- View a detailed list of **individual endpoints** that were scanned during the configuration scan.
- Navigate to the main **Configuration Inventory** by clicking **Configuration Scan Inventory**.
- **Export** the **Configuration Scan Inventor**y from the **Quantum Trust Hub**.

To do this:

1. Select the checkboxes corresponding to the records you want to export.
2. From the toolbar, by clicking **Export**.
3. From the **How would you like to download the data?** dialog box, select the file format for the exported configuration data.
4. Click **Submit**.

Quantum Readiness by Crypto Library



The **Quantum Readiness by Crypto Library** widget provides insight into the quantum security posture of the cryptographic libraries used across your environment. Cryptographic libraries implement the algorithms that protect your data and communications, so understanding their quantum readiness is essential for safeguarding against future quantum attacks.

The donut chart in this widget shows all the crypto libraries discovered in the scan and interactive legend lists the number of instances of usage of each library.

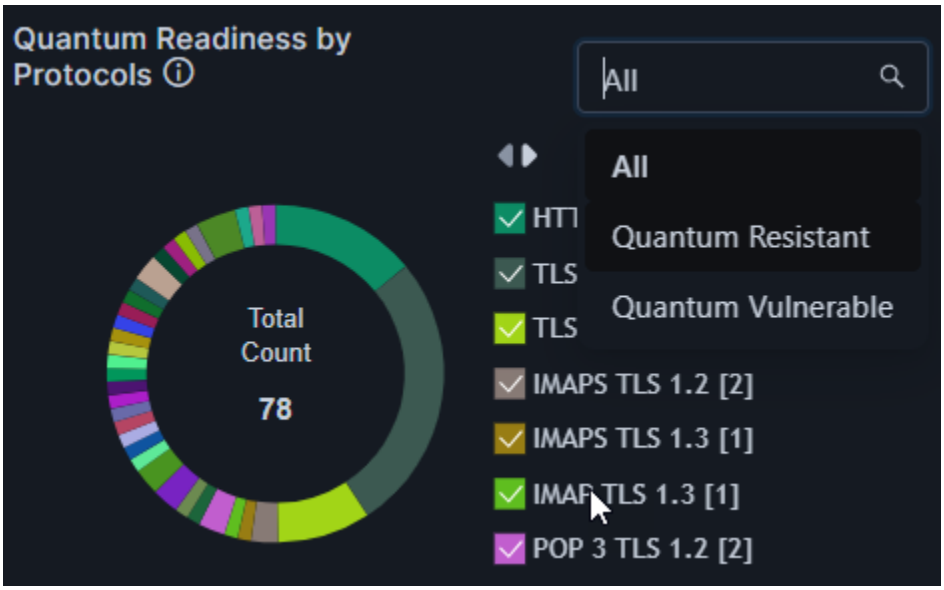
Use the interactive legends to update the visualization to see the usage distribution for only specific libraries.

Use the dropdown menu in the top-right corner of the widget to update the visualization for their quantum readiness status, from the following values:

Option	Description
All	Displays all cryptographic libraries detected, regardless of their quantum readiness status, giving a complete overview of your cryptographic landscape.
Quantum Resistant	Shows only cryptographic libraries that implement **post-quantum cryptographic algorithms**, ensuring strong resistance against quantum computing threats.
Quantum Vulnerable	Lists cryptographic libraries that rely on **classical cryptographic algorithms** vulnerable to quantum attacks, highlighting areas requiring immediate attention.

1. Select a filter from the dropdown menu to view cryptographic libraries by their quantum readiness status.
2. Identify vulnerable libraries and prioritize updates or replacements with quantum resistant versions.

Quantum Readiness by Protocols



The **Quantum Readiness by Protocols** widget gives you an overview of the quantum security status of cryptographic protocols in use within your environment. Protocols such as TLS, SSH, and others play a key role in securing communications, so assessing their readiness against quantum threats is crucial.

The donut shows the total number of protocols discovered by the scan and the usage distribution for the protocols. The interactive legend lists the protocols discovered along with the number of usage instances for each protocol.

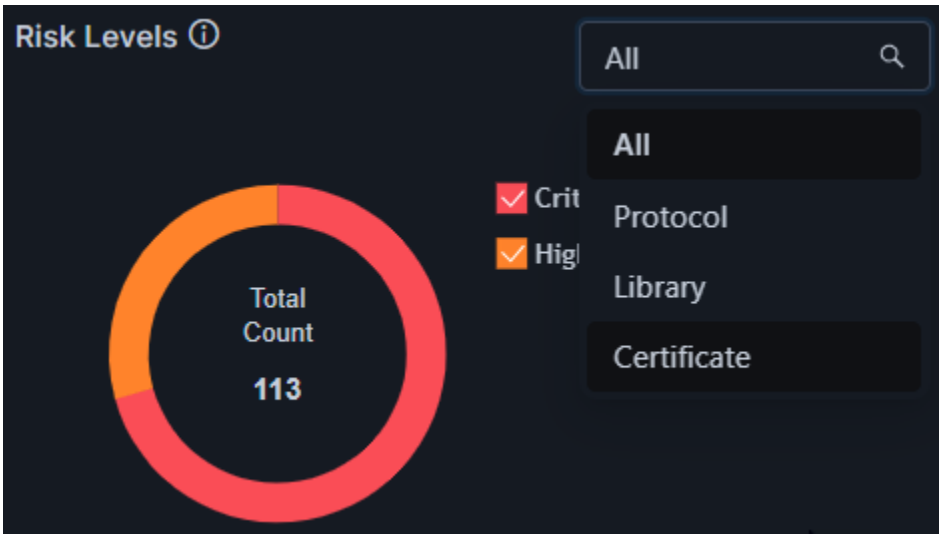
Use the interactive legends to update the visualization to see the usage distribution for only specific protocols.

Use the dropdown menu in the top-right corner of the widget to update the visualization for protocols based on their quantum readiness, from the following values:

Option	Description
All	Displays all cryptographic protocols detected, regardless of their quantum readiness status, for a complete picture of your security posture.
Quantum Resistant	Shows only protocols that use **quantum-safe algorithms** or configurations designed to resist attacks from quantum computers.
Quantum Vulnerable	Lists protocols that rely on **classical cryptographic methods vulnerable to quantum attacks** and should be prioritized for upgrade or replacement.

1. Select the required filter from the dropdown to focus on all protocols, only quantum resistant ones, or those that are quantum vulnerable.
2. Identify vulnerable protocols and understand which protocols in your environment are at risk from quantum attacks.

Risk Levels



The **Risk Levels** widget provides an at-a-glance summary of the quantum risk associated with different cryptographic components in your environment. It helps you understand where vulnerabilities exist and prioritize your remediation efforts.

The donut chart in this widget shows the total number of cryptographic components (protocols, libraries, and certificates) detected. It also shows the risk level distribution across these components. The chart

legend lists the risk level of the detected components and also indicates the number of components mapping to each risk level.

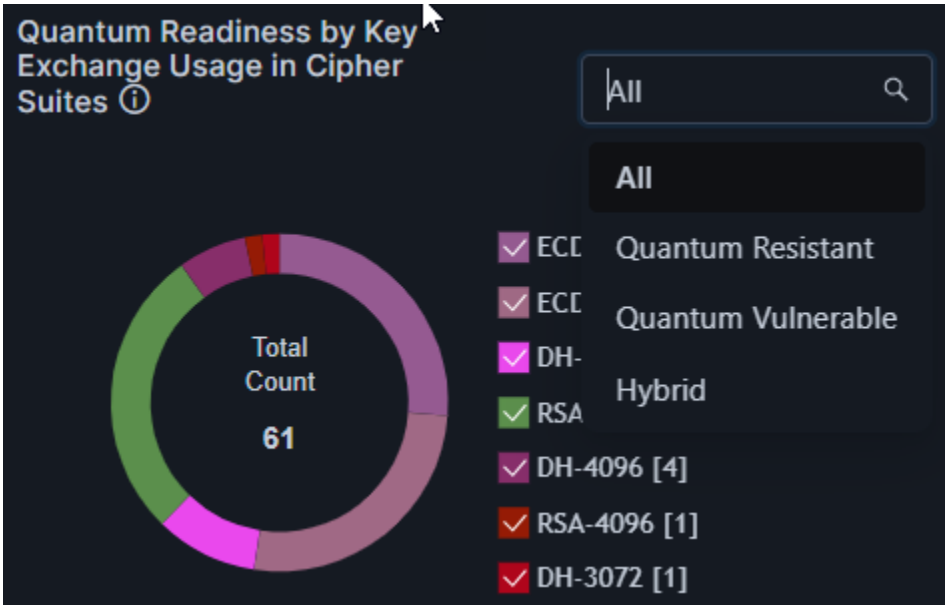
Use the interactive legends to filter the risk data for a specific risk level(s).

Use the dropdown menu to filter risk data by component type, from the following options:

Option	Description
All	Displays risk levels for **all cryptographic components** combined, including protocols, libraries, and certificates. Use this for a holistic view of your quantum risk exposure.
Protocol	Shows risk levels specifically related to **cryptographic protocols** (e.g., TLS versions, SSH), highlighting protocols that may be vulnerable to quantum attacks.
Library	Filters the view to show risk levels in **cryptographic libraries** or algorithms implemented within your systems, helping identify weak or deprecated crypto code.
Certificate	Focuses on the quantum risk posed by **digital certificates**, based on their encryption algorithms and PQC readiness. This helps spot certificates needing urgent replacement or upgrade.

1. Use the dropdown to select a category based on your focus area: protocols, libraries, certificates, or all combined.
2. Review the distribution of risk levels (e.g., **Critical, High, Medium, Low, Unknown**) displayed in the widget.

Quantum Readiness by Key Exchange Usage in Cipher Suites



The **Quantum Readiness by Key Exchange Usage in Cipher Suites** widget displays an analysis of the key exchange algorithms used within your cipher suites, focusing on their resilience against quantum computing threats.

Key exchange is a critical part of establishing secure communications, and this widget helps you understand how prepared your cryptographic configurations are for post-quantum security.

The donut chart on the widget shows the total number of key exchanges observed across all cipher suites. The legend lists all key exchange instances along with the individual count of occurrence.

Use the interactive legend to filter the visualization for specific key exchange instances.

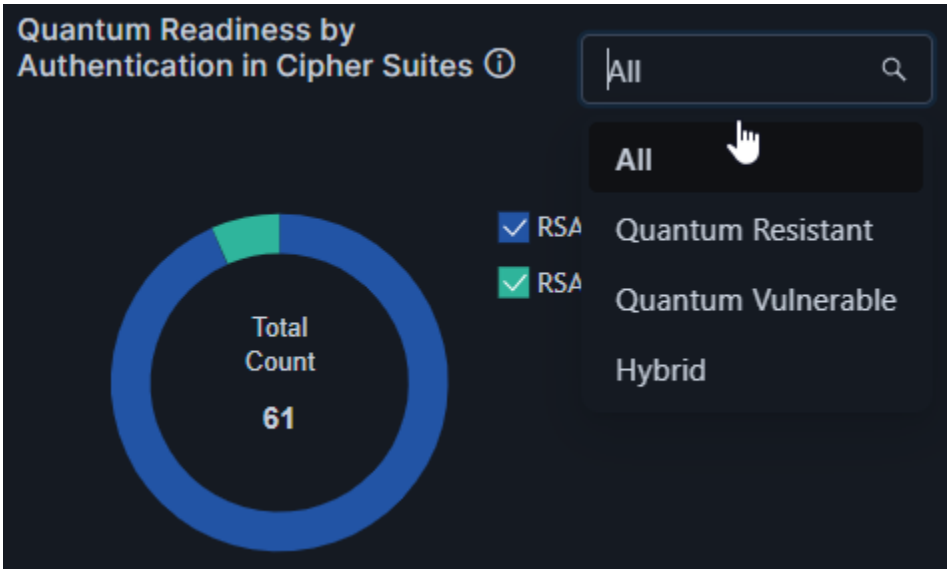
Use the dropdown menu to filter key exchange algorithms based on their quantum readiness status, for the following values:

Option	Description
All	Shows all key exchange algorithms detected in your cipher suites, regardless of their quantum readiness. Use this view to get a complete picture of your cryptographic posture.
Quantum Resistant	Displays only key exchange algorithms that are considered secure against quantum attacks, such as lattice-based or other post-quantum algorithms.
Quantum Vulnerable	Lists key exchange algorithms that rely on classical cryptography (e.g., Diffie-Hellman, ECDH) which can be broken by quantum computers and therefore require urgent upgrade.

Option	Description
Hybrid	Shows key exchange methods that combine classical and post-quantum algorithms, providing enhanced security during the transition to quantum-safe cryptography.

1. Select a filter from the dropdown menu to focus on specific key exchange algorithm types.
2. Identify vulnerable key exchange algorithms that should be prioritized for upgrade or replacement.

Quantum Readiness by Authentication in Cipher Suites



The **Quantum Readiness by Authentication in Cipher Suites** widget provides an overview of the cryptographic strength of your cipher suites based on their authentication mechanisms. This helps you understand how prepared your network communications are against quantum computing threats.

The widget shows the cryptographic algorithms/cipher suites used for authentication across your environment, as well as the total and individual count of usage. Use the interactive legend to filter the visualization for a specific cryptographic algorithm/cipher suite.

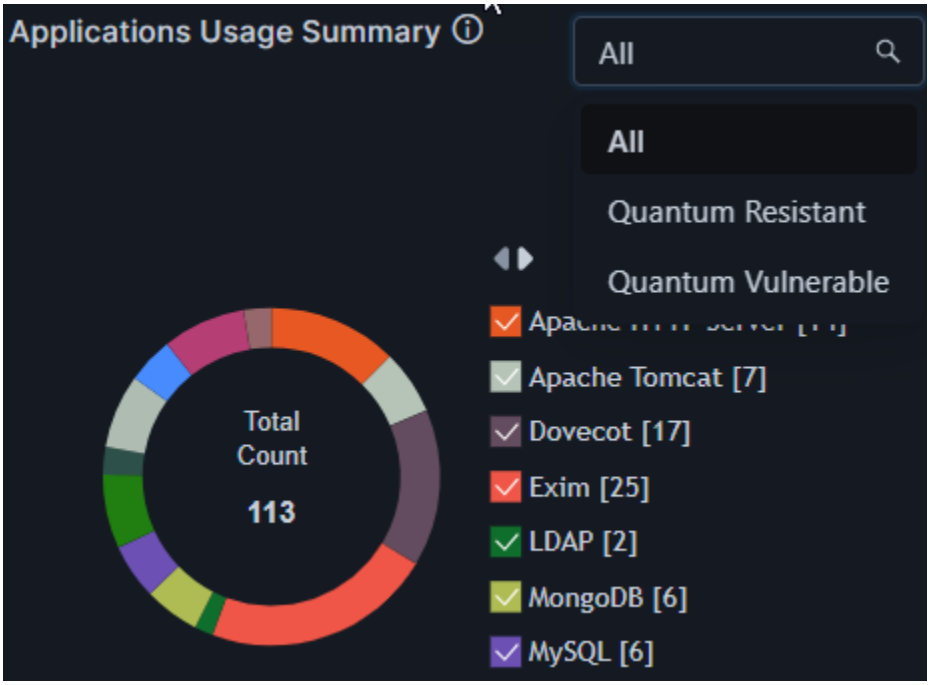
Use the dropdown menu to filter cipher suites according to their quantum readiness status:

Option	Description
All	Displays all cipher suites detected across your environment, regardless of their quantum resistance status. Use this for a comprehensive view.
Quantum Resistant	Shows only cipher suites that use **quantum-safe authentication algorithms**, offering strong protection against both classical and quantum attacks.

Option	Description
Quantum Vulnerable	Lists cipher suites that rely on **classical authentication methods vulnerable to quantum attacks** (e.g., RSA, ECDSA). These require prompt remediation.
Hybrid	Displays cipher suites that implement a **combination of classical and post-quantum algorithms** to enhance security during the transition to quantum-safe cryptography.

1. Use the dropdown to filter cipher suites by quantum readiness status.
2. Identify vulnerable cipher suites that require upgrading to post-quantum or hybrid authentication methods.

Applications Usage Summary



The **Applications Usage Summary** widget provides a quick visual summary of the cryptographic posture of your applications based on their usage of cryptographic configurations and certificates. This helps you assess how well your application environment is protected against quantum computing threats.

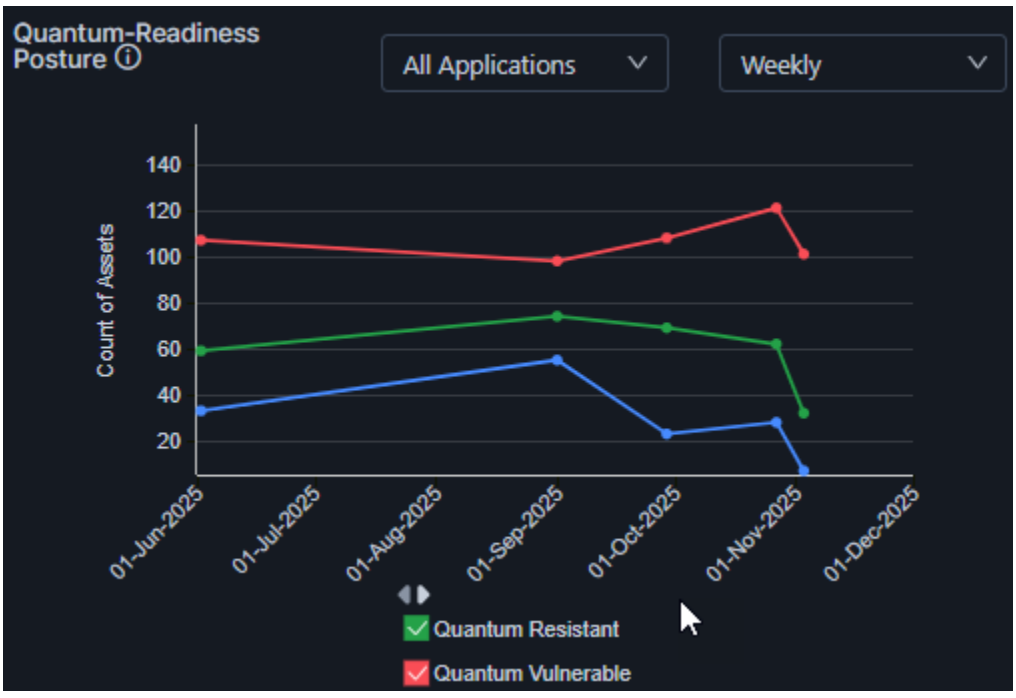
The widget shows the total number of applications detected in your environment. The legend lists these applications and also includes a count which indicates the number of cryptographic usages detected for that application. Use the interactive legend to filter the chart visualization for specific applications.

Use the dropdown menu to filter chart data based on the quantum readiness of the applications, from the following values:

Option	Description
All	Displays **all applications** in your environment, regardless of their cryptographic posture. Use this view to get a full inventory snapshot and identify areas for improvement.
Quantum Resistant	Shows only applications that are currently using **quantum-safe cryptographic configurations and/or certificates**. These applications are considered prepared to withstand future quantum attacks.
Quantum Vulnerable	Displays applications using **legacy or quantum vulnerable encryption** (e.g., RSA, ECC, SHA-1, TLS 1.0/1.1). These applications require **remediation** to avoid exposure to potential quantum threats.

1. Use the **dropdown filter** to switch between views (**All, Quantum Resistant, Quantum Vulnerable**) based on your focus area.
2. Identify and prioritize **vulnerable applications** for cryptographic upgrades.

Quantum-Readiness Posture



The **Quantum-Readiness Posture** chart displays a holistic view of your organization's readiness for a PQC transition for a selected period of duration.

It is essentially the configuration count data represented graphically. The widget offers the additional ability to view trends for specific applications over a specific duration by selecting the required values from the dropdown lists in the top-right corner of the widget.

The chart legends are interactive. Select/clear the checkbox for a risk severity level to show/hide, respectively, the corresponding data on the chart.

Configuration Scan Inventory

To access the configuration scan inventory:

Prerequisite: Verify that your user role has the required **ACF permission** to view configuration scan inventory. To enable ACF permission, click here.

1. To view the Configuration Scan inventory, go to **Menu > Quantum Trust Hub > Inventory > Configuration Scan**.

   You will be redirected to the **Configuration Scan Inventory**.

   The **Configuration Scan Inventory** provides a comprehensive, category-wise view of all cryptographic configurations across your organization's IT infrastructure - including protocols, cipher suites, encryption algorithms, and security settings used across applications, servers, and network devices.

   This inventory is critical for evaluating your system's quantum readiness from a configuration perspective, ensuring that all components adhere to evolving post-quantum cryptographic standards.

• Configuration Scan Inventory

Configuration Scan Inventory

The Configuration Scan Inventory page displays the list of scanned assets and their cryptographic configuration details. This report helps security analysts evaluate the encryption protocols, cipher suites, and quantum-readiness of services running within the network. It is commonly used to verify compliance with cryptographic standards, detect weak configurations, and ensure readiness for Post-Quantum Cryptography.

Verify that your user role has the required **ACF permission** to view configuration scan inventory. To enable ACF permission, click here.

Viewing the Configuration Scan Inventory

To view the configuration scan inventory:

To view the configuration scan inventory, go to **Menu > Quantum Trust Hub > Inventory > Configuration**.
You will be redirected to the **Configuration Scan Inventory**.

Common Inventory Functions

The table below explains the inventory functions for Configuration Scan Inventory.

Feature	Description
Filters	To filter the inventory for viewing specific data: 1. From one or more of the following dropdown lists, select the required filtering criteria: • **IP address** • **Quantum Readiness** • **Severity** • **Crypto Category** 2. Click **Apply**.
Search	Enter free text or keywords to search for specific entries in the inventory.
Export	To export the inventory data: 1. Select at least one record from the inventory to export the corresponding data. 2. From the menu bar, click **Export**. 3. From the **How would you like to download the data?** Dialog box, select your preferred export file format (**CSV** or **XLS**). 4. Click **Submit**. The inventory data is downloaded to your local system as a zipped file.
Pagination	Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory.

Feature	Description
Pagination Navigation	Use the pagination navigation buttons to move between the pages in the inventory.
Refresh	Use the **Refresh** button to reload the inventory to display the up-to-date records.

Configuration Scan Inventory

The configuration scan inventory displays the following field details to view the PQC readiness:

Column descriptions for the Configuration Scan Inventory page

Column Name	Description
IP address	Displays the IP address of the scanned host. Each row represents one detected service on a unique IP.
FQDN	Shows the domain name associated with the IP address. Useful for identifying hostnames in DNS-based scans.
Applications	Identifies the detected application or service running on the host (for example, exim, nginx, or apache).
Port	Specifies the network port used by the application. This indicates where the service is accessible.
Service Binding / Hostname	Displays the; • network binding format (IP:Port) that shows which address and port combination the service listens on • configured name/domain for that service
Crypto Category	Defines the type of cryptographic setting being reported (for example, Protocol, Cipher, Certificate, or Algorithm).
Crypto Value	Shows the protocol version or cryptographic mechanism in use (for example, TLS 1.3, SSL 3.0, etc.).
Cipher Suite	Lists the exact cipher suite negotiated for the TLS/SSL connection (for example, TLS_AKE_WITH_...).

Column descriptions for the Configuration Scan Inventory page (continued)

Column Name	Description
Key Exchange Algorithm	Indicates cryptographic algorithm used to securely negotiate encryption keys between parties during the initial phase of a secure communication session (e.g., TLS handshake).
Authentication	Specifies the authentication algorithm (for example, ECDSA, Dilithium, etc.) used for validating the identity of the communicating entities.
Severity	Displays the security impact level associated with the detected configuration. Levels may include Low, Medium, High, or Critical.
Quantum Readiness	Indicates whether the cryptographic configuration is resistant to quantum-based attacks.
Recommended Action	Provides guidance or next steps for remediation or optimization. If no action is required, it may display N/A.

Chapter 9: Configuring PQC Readiness/Post-Quantum Policies

Policies define the cryptographic standards and algorithms used to assess your environment's readiness for post-quantum cryptography (PQC). A PQC policy lets you establish a cryptographic governance framework that:

- Continuously evaluates your organization's algorithms and protocols against the PQC-readiness standards
- Automatically transforms non-compliant cryptographic elements into compliant ones based on organization-specific policy decisions

It is like your organization's quantum-readiness rulebook that is used for:

- Detecting all cryptographic algorithms/protocols in use across applications, infrastructure, and source code
- Classifying each algorithm as quantum Vulnerable or Quantum Resistant per custom PQC policy
- Maintaining the auditability of decisions and changes for security and compliance reporting
- Supporting crypto-agility so policies can evolve with new PQC standards or threat intelligence

AppViewX lets you create custom policies based on your organization's standards and requirements to override the default standards defined by NIST and the Grover/Shor algorithm. In the following sections, you will be walked through how AppViewX's Quantum Trust Hub lets you create custom code, certificate, and configuration policies, modify and delete the policies as per revisions in your requirements, and control policy enforcement.

> **Important:** Overrides do not alter the NIST standards or an algorithm's quantum resistance derived from Grover/Shor analysis. They only affect your organization's PQC score and Quantum Readiness reporting.

Viewing the PQC Policy Inventory

All PQC policies (code, configuration, and certificate) created by your organization are listed together in the main PQC policy inventory.

Prerequisite: Verify that your user role has the required **ACF permission** to view policy inventory. To enable ACF permission, click here.

To view the PQC Policy inventory, go to **Menu** > **Quantum Trust Hub** > **Policy**.

The **PQC Policy** page is displayed.

This page documents all key details related to each governance policy created for your organization's quantum-readiness framework. It also lets you create, modify, and delete policies, and control policy enforcement.

Understanding the PQC Policy Inventory

Fields	Description
Toolbar	The PQC policy inventory toolbar has the following options: • **Search**: Enter free text or keywords to search for specific policies in the inventory. • **Create**: To create a new policy (code, configuration, or certificate), click **Create**. For instructions, see Creating a Policy. • **Delete**: To delete a policy, click  . For instructions, see Deleting a Policy. • **Know More**: To read on policies and the types of policies AppViewX supports for its PQC implementation, click  . • **Pagination**: Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory. • **Pagination navigation**: Use the pagination navigation buttons to move between the pages in the inventory. • **Refresh**: Use the Refresh button to reload the inventory to display the up-to-date records.
Policy Name	User-assigned policy name
Description	Additional details related to the policy, if and as specified by the user
Policy Scope	Cryptographic asset that the policy was created for (**code**, **configuration**, **certificate**)
Policy Enforcement	This field controls enabling/disabling a policy. For instructions and ACF rules, see Enforcing a Policy.

Fields	Description
	Important: There can be only one active policy at a time.

Creating a Policy

This section of the product lets you create custom policies that can target one or all of the cryptographic assets in your infrastructure, to validate them for quantum-readiness and vulnerabilities.

Prerequisite: Verify that your user role has the required **ACF permission** to create policies. To enable ACF permission, click here.

Creating a Custom Policy

1. Go to **(Menu)** > **Qauntum Trust Hub** > **Policy**.
   The **Post-Quantum Policy** page is displayed, which is your complete policy inventory.
2. From the toolbar, click **Create**.
   The **Post-Quantum Policy > Create** page is displayed.
3. Under **Crypto Policy Management**:

   a. In the **Policy** field (mandatory), enter a descriptive name for identifying your policy.
   Constraints: Spaces and leading special characters are considered invalid for the policy name.

   b. In the **Description** field, enter additional details related to the policy.
   This could include the purpose of the policy, how it applies, and so on.

   c. Add rules to the policy targeting all or only specific cryptographic assets.
   - To add a code-related rule to the policy, go to the **Code** tab.

     For instructions, see Adding a Code Rule to Your Custom Policy.
   - To add a configuration-related rule to the policy, go to the **Configuration** tab.

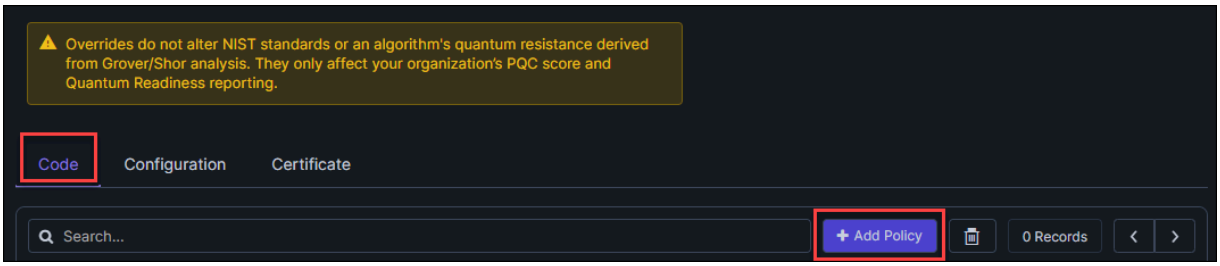
     For instructions, see Adding a Configuration Rule to Your Custom Policy.
   - To add a certificate-related rule to the policy, go to the **Certificate** tab.

     For instructions, see Adding a Certificate Rule to Your Custom Policy.

4. Click **Save**.
   The policy is created and listed in the PQC policy inventory.

Adding a Code Rule to Your Custom Policy

1. From the scope toolbar, click **Code** and then click **Add Policy**.



   The **Add Policy** pop-up dialog box is displayed.

2. Enter/Select the following details for the code rule in your custom policy:

> **Important:** Overrides do not alter NIST standards or an algorithm's quantum resistance derived from Grover/Shor analysis. They only affect your organization's PQC score and Quantum Readiness reporting.

Fields	Description
***Type**	To override the default quantum-safety status for an encryption algorithm, from the dropdown list, select the required algorithm type.
***Override Classification**	From the dropdown list, select the new quantum-safety status value for the selected algorithm, which will override its default value.
***Key type & strength**	From the dropdown list, select the new key type and strength that will override the algorithm's default values.
Notes	Enter your justification for the override configured using the above fields. While this is an optional field, entering the description is a recommended practice to ensure a knowledge base to guide decisions for future configurations to a policy.
*: *Mandatory fields*	

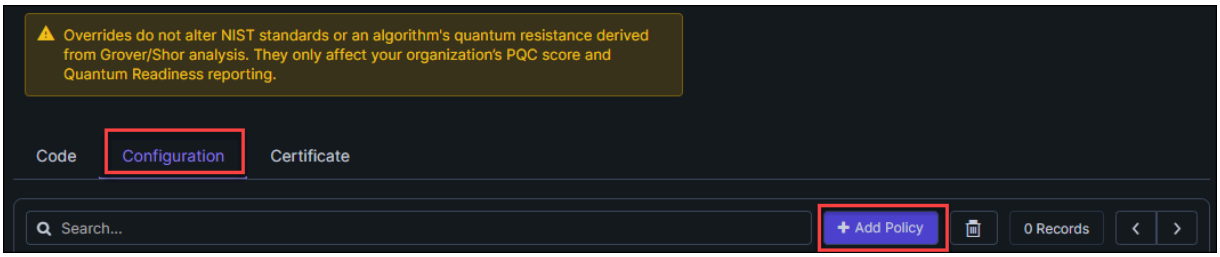
3. Click **Add Policy Rule**.

   The code rule is added to the rule inventory.

   To read on the details in the rule inventory, click here.

Adding a Configuration Rule to Your Custom Policy

1. From the scope toolbar, click **Configuration** and then click **Add Policy**.



   The **Add Policy** pop-up dialog box is displayed.
2. Enter/Select the following details for the configuration rule in your custom policy:

> **Important:** Overrides do not alter NIST standards or an algorithm's quantum resistance derived from Grover/Shor analysis. They only affect your organization's PQC score and Quantum Readiness reporting.

Fields	Description
***Type**	To override its default NIST classification, from the dropdown list, select the required protocol or cipher suite component.
***Override Classification**	From the dropdown list, select the new quantum-safety status value for the selected protocol/cipher suite component, which will override its default value.
***Key type & strength**	From the dropdown list, select the key type and strength that will override the selected protocol/cipher suite component's default values.
Notes	Enter your justification for the override configured using the above fields. While this is an optional field, entering the description is a recommended practice to ensure a knowledge base to guide decisions for future configurations to a policy.
*: *Mandatory fields*	

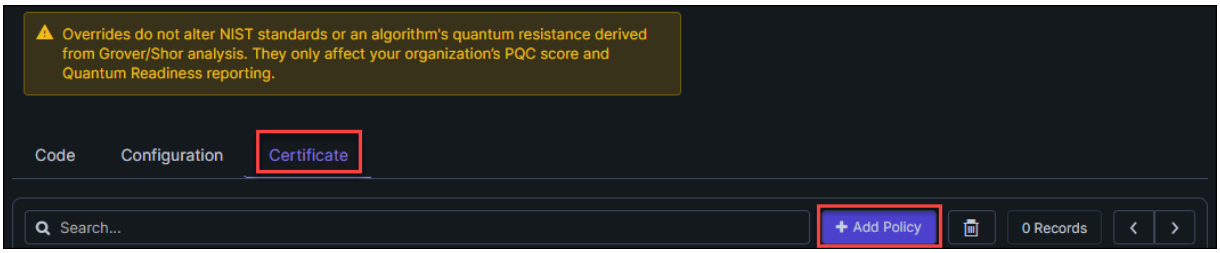
3. Click **Add Policy Rule**.

The configuration rule is added to the rule inventory.

To read on the details in the rule inventory, click here.

Adding a Certificate Rule to Your Custom Policy

1. From the scope toolbar, click **Certificate** and then click **Add Policy**.



   The **Add Policy** pop-up dialog box is displayed.
2. Enter/Select the following details for the certificate rule in your custom policy:

> **Important:** Overrides do not alter NIST standards or an algorithm's quantum resistance derived from Grover/Shor analysis. They only affect your organization's PQC score and Quantum Readiness reporting.

Fields	Description
***Type**	To override the default quantum-safety status for an encryption algorithm, from the dropdown list, select the required algorithm type.
***Override Classification**	From the dropdown list, select the new quantum-safety status value for the selected algorithm, which will override its default value.
***Key type & strength**	From the dropdown list, select the key type and strength that will override the selected algorithm's default values.
Notes	Enter your justification for the override configured using the above fields. While this is an optional field, entering the description is a recommended practice to ensure a knowledge base to guide decisions for future configurations to a policy.
*: *Mandatory fields*	

3. Click **Add Policy Rule**.

The certificate rule is added to the rule inventory.

To read on the details in the rule inventory, click here.

Understanding the Rule Inventory

The rule inventory for each cryptographic asset (code, configuration, and certificate) is displayed in the corresponding tab on the toolbar.

Common Inventory Functions

Fields	Description
Search	Enter free text or keywords to search for specific policies in the inventory.
	To delete a rule from the inventory, select the corresponding checkbox and click  .
Pagination	Use the pagination control dropdown to select the number of records that will be displayed per page of the inventory. You can select to display 25, 50, 75, or 100 records per page of the inventory.
Pagination navigation	Use the pagination navigation buttons to move between the pages in the inventory.

Rule Details

Fields	Description
Type	Algorithm/protocol/cipher suite component for which the quantum-status classification has been modified
Key Type & Strength	Default key type and strength of the selected algorithm/protocol/cipher suite component
Default Quantum Status	Default quantum-status of the selected algorithm/protocol/cipher suite component
Organization override	New quantum-status classification assigned to the selected algorithm/ protocol/cipher suite component, which will override the default value
Added By	Name of the user who created the policy rule

Fields	Description
Date	Date on which the policy rule was created

Enforcing a Policy

At a time, only one active policy can govern the PQC-focused scans for quantum-readiness. The admin user can enable/disable a policy, to activate the required policy.

Enabling ACF for Policy Enforcement

Note: To learn how to enable the ACF permission for the user roles to access the **Policy** under **Quantum Trust Hub**, click here.

Enabling/Disabling a Policy

To enable a policy, from the Policy Enforcement column in the PQC Policy inventory, turn on the toggle key.

Note: You will not be able to enable a policy unless the active policy has been disabled.

To disable a policy, from the Policy Enforcement column in the PQC Policy inventory, turn off the toggle key.

Important: When a policy is enabled, the currently enabled policy is automatically disabled. Before disabling a policy, it is mandatory that at least one other policy is enabled. All policies cannot be disabled. If this is attempted the message **Disabling this policy requires enabling another applicable policy.** is displayed.

Modifying a Policy

Important: The default policy cannot be modified.

Prerequisite: Verify that your user role has the required **ACF permission** to modify policies. To enable ACF permission, click here.

1. From the PQC Policy inventory, click the **Policy Name** of the policy that has to be modified.
   Policy details entered at the time of policy creation are displayed.

2. Update the policy details as required.
   For field descriptions, see the corresponding instruction in Creating a Policy.
3. Click **Save**.
4. Click **Update**.
   A confirmation message is displayed to indicate if the policy update was a success or a failure.

   If the policy update is a success, all reports are updated immediately according to the modifications made.

Deleting a Policy

Important: The default policy cannot be deleted.

Prerequisite: Verify that your user role has the required **ACF permission** to delete policies. To enable ACF permission, click here.

1. From the PQC Policy inventory, select the checkbox corresponding to the policy you want to delete. You can select more than one policy.
2. From the toolbar, click **Delete**.
   The selected policy is/policies are deleted.

Chapter 10: Quantum Solution Reference Guides

- For Certificate Scan Outcome Analysis
- For Configuration Scan Outcome Analysis
- For Code Scan Outcome Analysis

For Certificate Scan Outcome Analysis

Scope of Discovery

What it shows:

- How certificates were discovered:
  - Agentless network scan (TLS handshake analysis)
  - Agent-based scan (local stores/configs on servers)
  - Managed device & CA integrations (PKI, load balancers, appliances).

Why it matters:

- Ensures you have complete coverage across public and internal environments
- Prevents hidden certificates from being left out of PQC planning.

Action:

- Verify that all discovery sources are enabled to avoid blind spots.

Certificate Inventory

What it shows:

- Full certificate chain elements:
  - **Root** (trust anchors)
  - **Intermediate** (delegated issuers)
  - **Device/Service** (bound to servers, apps, devices)
  - **Leaf** (end-entity certs for TLS, S/MIME, code signing, etc.)
- Classification: **public-facing vs internal**.

Why it matters:

- PQC readiness is impacted not only by leaf certs, but also by **roots and intermediates**
- A weak root/intermediate undermines the trust of all dependent certificates.

Action:

- Review inventory to ensure all trust chain elements are scanned.

Algorithms and Key Strength

What it shows:

- Breakdown of:
  - **Asymmetric algorithms** (RSA, ECC, PQC, Hybrid)
  - **Hash/signature algorithms** (SHA-1, SHA-2, SHA-3, Dilithium, Falcon etc)
  - **Key sizes** (RSA-2048/4096, ECC-P256/P384, PQC parameter sets).

Why it matters:

- Quantum computers will break RSA and ECC
- Deprecated algorithms (MD5, SHA-1) are already insecure today
- PQC algorithms (ML-KEM,ML-DSA,SLH-DSA) align with NIST recommendations.

Action:

- Identify certificates using weak or quantum-vulnerable algorithms
- Prioritize replacing them with hybrid or PQC-ready certificates.

PQC Score Calculator

The PQC Score Calculator for Certificates measures the quantum resilience of digital certificates by analyzing their cryptographic algorithms, key strengths, and configurations.It helps identify which certificates are quantum-safe, hybrid, or vulnerable to future quantum attacks by assessing the algorithm type, key size, and overall cryptographic strength

Using standardized criteria based on NIST PQC guidelines and quantum threat models (such as Shor's and Grover's algorithms), the calculator assigns a numerical score that reflects the level of readiness for post-quantum security.

Type	Score
Quantum-Resistant certificates	1

Type	Score
Hybrid certificates	0.6
Quantum-Vulnerable certificates	0

PQC Score Formula

PQC Score = ∑(Count of Score) / (Total Certificates) X *10

Explanation

- **∑(Count of Score)** → The sum of all individual certificate scores (for example, 0 + 12 + 10 = 22)
- **Total Certificates** → The total number of certificates analyzed (e.g., 100)
- The result is multiplied by **10** to convert it into a **1–10 scale**

Eg: Assume you have 100 certificates

- 70 Quantum-Vulnerable→ 70 × 0 = 0
- 20 hybrid → 20 × 0.6 = 12
- 10 Quantum-Resistant→ 10 × 1 = 10

Example calculation

PQC Score = (22/100) * 10 = 2.2

Threat Level Interpretation

Severity Assessment (NIST-Aligned)

What it shows:

- Certificates rated by risk level.

Cert Type	Description	Certificate Validity Window	NIST PQC Timeline	Severity
Classical: RSA ≤ 2048 bits	~112-bit classical strength	After 2030	Insecure beyond 2030	Critical

Cert Type	Description	Certificate Validity Window	NIST PQC Timeline	Severity
Classical: RSA ≤ 2048 bits	~112-bit classical strength	On/before 2030	Allowed within window	High
Classical: RSA 2048–3071 bits	~112–128-bit classical strength	After 2035	Insecure beyond 2035	Critical
Classical: RSA 2048–3071 bits	~112–128-bit classical strength	On/before 2035	Allowed within window	High
Classical: RSA ≥ 3072 bits	≥128-bit classical strength	Any	Classical secure	High
Classical: DSA ≤ 1024 bits	Weak / legacy	Any	Already insecure	Critical
Classical: DSA ≤2048 bits	Moderate classical strength	Any	Classical secure	High
Classical: DSA ≥ 3072 bits	Strong classical strength	Any	Classical secure	High
Classical: ECC < P-256	Weak curve	After 2030	Insecure beyond 2030	Critical
Classical: ECC < P-256	Weak curve	On/before 2030	Allowed within window	High
Classical: ECC ≥ P-256	Modern curve	Any	Classical secure	High
Classical: ECC P-384 / P-521	Strong curve	Any	Classical secure	High
Classical: Ed25519 / Ed448	Modern curve	Any	Classical secure	High
Hybrid: RSA + PQC	Classical + PQC	Any	Recommended now → post-2035	Medium
Hybrid: ECC + PQC	Classical + PQC	Any	Recommended now → post-2035	Medium

Cert Type	Description	Certificate Validity Window	NIST PQC Timeline	Severity
Pure PQC	ML-KEM ML-DSA SLH-DSA	Any	Recommended now → post-2035	Low (Quantum Safe)

Why it matters:

- Helps you prioritize migration efforts
- Aligns with **NIST PQC transition guidance**.

Action:

- Remediate **Critical** and **High** first (public-facing certificates first).

Key Usage and Extended Key Usage (EKU)

What it shows:

- **Key Usage:** Digital Signature, Key Encipherment, Data Encipherment, Certificate Signing
- **Extended Key Usage (EKU):** TLS Web Server, Email Protection, Code Signing, Client Authentication.

Why it matters:

- Different usages have different business impact

  Example: PQC vulnerability in code signing certs may be higher risk than in a test server cert.

Action:

- Focus migration on business-critical usages (e.g., TLS for customer apps, email protection, code signing).

Quantum Readiness Classification

What it shows:

- **Classical** → Quantum-vulnerable
- **Hybrid** → Transitional (RSA/ECC + PQC)
- **PQC-only** → Quantum-safe.

Why it matters:

- Gives a snapshot of your organization's post-quantum maturity.

Action:

- Track progress from **Classical** → **Hybrid** → **PQC only** certificates.

High-Risk Certificates

What it shows:

- Certificates using **deprecated algorithms** (MD5, SHA-1, RSA-1024, weak ECC curves)
- Public vs internal exposure breakdown
- Risk by Trust Hierarchy: Evaluates each level of the certificate chain.
  1. Root CAs → If weak, all downstream certs are at risk.
  2. Intermediate CAs → If vulnerable, compromise impacts every leaf they issue.
  3. Leaf Certificates → Endpoint exposure, typically seen in TLS handshakes.

Why it matters:

- Public facing weak certificates pose the highest attack risk
- A weak intermediate or root introduces risk across hundreds or thousands of certificates
- Trust hierarchy risk helps prioritize remediation not just by usage, but by chain impact.

Action:

- Replace or re-issue from secure intermediates and roots first
- Ensure PKI trust anchors (roots & intermediates) are PQC-aligned
- Prioritize migration of public facing chains over internal one.

Quantum Readiness Mapping

What it shows:

- Certificates checked against:
  - **NIST PQC guidance**
  - Internal crypto/security policies
  - Standards (PCI-DSS, HIPAA, eIDAS, ETSI).

Dashboards

What it shows:

- **PQC Readiness Score:** % PQC-ready vs vulnerable certificates
- **Classification View:** Classical, Hybrid, PQC-only
- **Quantum Readiness Trend:** Progress over time
- **Public vs Internal Risk:** PQC exposure split
- **Algorithm Summary:** Usage of asymmetric, hash, signature algorithms
- **Key Usage & EKU Summary:** PQC readiness mapped to business functions
- **Risk by Trust Hierarchy:** Risk summarized by root, intermediate and leaf certificates.

Why it matters:

- Provides **executive and technical visibility** into PQC progress.

Action:

- Track trends over time to demonstrate migration progress.

Migration Guidance

What it shows:

- Recommended steps for moving to PQC
  - Replace weak certificates with hybrid/PQC-ready certs
  - Update PKI/CA infrastructure for PQC issuance
  - Enforce minimum key sizes and NIST-approved PQC algorithms.

Why it matters:

- Ensures a structured, low-risk migration to PQC without disrupting services.

Action:

- Follow a phased approach: **Classical** → **Hybrid** → **PQC** only.

For Configuration Scan Outcome Analysis

This guide provides a detailed overview of the Configuration Quantum Readiness Report, explaining each section and offering guidance on how to interpret the results. It focuses on identifying cryptographic protocols, libraries, ciphers, and bound certificates that may affect your organization's readiness to transition to post-quantum cryptography.

Scope of Discovery

The configuration scan operates using an **agent-based model**:

- **Agent Deployment:** A lightweight agent is installed on servers or application hosts
- **Local Discovery:** The agent inspects system configuration files, SSL/TLS settings, crypto libraries, and certificate bindings locally
- **Full Visibility:** This ensures discovery of all protocols, libraries, and ciphers in use as well as certificates actively bound to services
- **Accuracy:** Because discovery is performed locally, the scan is not limited by network visibility and cannot be bypassed by load balancers or intermediate proxies.

Protocols

What it shows:

All SSL/TLS protocols used by servers running in the endpoint.

Why visibility into all configured protocols is required for PQC migration?

Even if system is configured to prefer TLS 1.2 or TLS 1.3, legacy protocols like SSLv2, SSLv3, TLS 1.0, and TLS 1.1 remain exploitable if they are still enabled because

- **Downgrade attacks**
  - Attackers can trick a client and server into falling back to a weaker protocol that both still support.
- **Negotiation flexibility**
  - Some older clients may still attempt connections using weak protocols
  - If accepted, these connections are insecure, exposing data and credentials.
- **Known cryptographic flaws**
  - SSLv2/SSLv3/TLS 1.0/1.1 use broken primitives (e.g., RC4, MD5, weak MACs) that can be exploited regardless of preference order.

- **Quantum readiness impact**
  - Legacy protocols cannot support modern cipher suites or PQC-hybrid extensions, making them blockers for future migration.

Action:

- Disable SSLv2, SSLv3, TLS 1.0, and TLS 1.1
- Require TLS 1.2 minimum; prefer TLS 1.3
- Validate vendor support for PQC enabled TLS extensions
- Ensure cipher suites under each protocol are modern (Eg:AES-GCM/ChaCha20) with forward secrecy.

Libraries

What it shows:

All cryptographic libraries and frameworks used by the supported servers (e.g., OpenSSL, BouncyCastle, WolfSSL, JCE), including their versions.

Why it matters:

Unsupported or outdated libraries introduce vulnerabilities and block Post Quantum Cryptographic adoption. Only modern libraries will integrate NIST-standard PQC algorithms.

Action:

- Upgrade to the latest supported versions
- Standardize libraries across environments
- Select libraries with active PQC support or published migration roadmaps.

Ciphers

What it shows:

All cipher suites defined in the SSL/TLS configuration that are in use.

Why visibility into all configured options is required for PQC migration?

Even if a system prefers strong modern algorithms, the full list of configured options still matters, because

- **Downgrade attacks**
  - If weak or legacy ciphers/protocols (e.g., RSA key exchange, 3DES, TLS 1.0) remain enabled, an attacker can force a fallback to them bypassing the “preferred” PQC-ready choice.
- **Quantum readiness depends on availability, not preference**

  - PQC migration planning requires knowing which algorithms and key exchanges are actually allowed, not just which ones are typically used in negotiation
  - If classical-only suites are still configured, the environment is not quantum-ready.
- **Consistency across environments**
  - Inconsistent configurations between servers, services, and applications can lead to some endpoints being PQC-ready while others remain vulnerable
  - Discovery of all configured options ensures alignment.
- **Audit and compliance requirements**
  - Regulators and auditors expect proof that legacy/quantum-vulnerable algorithms are fully disabled, not just unused.

Action:

- Remove weak/deprecated ciphers
- Enforce forward-secrecy suites (ECDHE + AES-GCM/ChaCha20)
- Prepare migration strategy for PQC-hybrid suites once standardized.

How Cipher Suites are grouped for PQC readiness evaluation

When the system scans a server (for example, an Apache web server), it identifies all the cipher suites the server supports under protocols like TLS 1.2 or TLS 1.3.

These cipher suites determine how secure communication is established, including how keys are exchanged, how identities are verified, and how data is encrypted.

Example: Cipher Suites Retrieved from an Apache Server

When a PQC discovery scan runs against an Apache endpoint, the system might retrieve a list like this for each service binding:

Cipher Suite	Key Exchange	Authentication
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	DH	RSA
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	DHE	RSA
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	ECDH	RSA
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE	RSA
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE	ECDSA
TLS_AES_256_GCM_SHA384	ECDHE	ML-DSA

Cipher Suite	Key Exchange	Authentication
TLS_CHACHA20_POLY1305_SHA256	ML-KEM	RSA
TLS_AES_256_GCM_SHA384	ML-KEM	ML-DSA
TLS_DH_anon_WITH_AES_128_CBC_SHA	DH	None

This list shows a mix of classical and PQC configurations, which the platform then analyzes and groups based on their cryptographic properties.

Step 1: Discover Cipher Suites

Every discovered cipher suite contains three building blocks

1. **Key Exchange:** how systems agree on a shared secret key.
2. **Authentication:** how the server (or client) proves its identity.
3. **Encryption:** how data is protected after the handshake.
4. **Hashing:** how data integrity is ensured.

The scanner extracts the first two because they determine quantum vulnerability.

Examples from the Apache scan:

• TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 → DHE (key exchange), RSA (auth)
• TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 → ECDHE (key exchange), ECDSA (auth)
• TLS_AES_256_GCM_SHA384 → ML-KEM (key exchange), ML-DSA (auth).

Step 2: Classify Algorithms by Type

Each algorithm is classified as **Classical**, **Hybrid**, or **PQC** based on its cryptographic foundation.

Category	Example Algorithms
Classical	DH, DHE, ECDH, ECDHE, RSA, ECDSA
Hybrid	ECDHE + ML-DSA, X25519MLKEM768
PQC	ML-KEM, SLH-DSA, ML-DSA(signatures)

Step 3: Apply Grouping Logic

Grouping removes redundancy and focuses on unique cryptographic behavior

Condition	Grouping Behavior in inventory	Details
Same key exchange + same authentication	Grouped together	Both handshake stages are identical & same cryptographic behavior
Same key exchange, but different authentication	Separate groups	Authentication uses different signing algorithms
Different key exchanges, even with same authentication	Separate groups	Key generation and mathematical base differ
PQC or hybrid key exchanges (Kyber, Kyber+RSA)	Separate groups	Represents distinct cryptographic families
Anonymous (no authentication)	Separate group	Insecure

Step 4: Grouped Results in the Inventory

After grouping, the Apache example appears as:

Protocol	Key Exchange Group	Authentication	Cipher Suites Included in Group
TLS 1.3	DH	RSA	TLS_DH_RSA_WITH_AES_256_GCM_SHA384
TLS 1.3	DHE	RSA	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS 1.3	ECDH	RSA	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS 1.3	ECDHE	RSA	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS 1.3	ECDHE	ECDSA	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS 1.3	ECDHE	ML-DSA	TLS_AES_256_GCM_SHA384
TLS 1.3	ML-KEM	RSA	TLS_CHACHA20_POLY1305_SHA256
TLS 1.3	ML-KEM	ML-DSA	TLS_AES_256_GCM_SHA384
TLS 1.3	DH (Anonymous)	None	TLS_DH_anon_WITH_AES_128_CBC_SHA

Step 5: Why Grouping Happens

Grouping cipher suites isn’t just about simplifying the report, it's about ensuring accuracy, clarity, and meaningful analysis of your cryptographic posture.

Each cipher suite defines a unique combination of

- **Key Exchange:** how the encryption key is negotiated
- **Authentication:** how the endpoint's identity is verified.

Grouping ensures that suites with identical handshake behavior and equal PQC risk are treated as a single entry, rather than being repeated multiple times.

To Represent Unique Cryptographic Behavior

Every key exchange and authentication pairing represents a distinct handshake pattern.

By grouping cipher suites with the same key exchange and same authentication, the system ensures that:

- Each group reflects a unique mathematical process, and
- No unrelated ciphers are mixed together.

For example:

- All ECDHE_RSA ciphers perform the exact same key negotiation and identity check → grouped together but DHE_RSA or ECDHE_ECDSA differ in key exchange or authentication → shown separately.

This makes each row in the PQC inventory represent one precise cryptographic combination.

To Maintain Quantum Risk Accuracy

Even small variations in cipher structure (like DH vs ECDHE or RSA vs ECDSA) affect how they'll perform against quantum attacks.

Grouping only by **identical key exchange and authentication** ensures the PQC readiness score is accurate

- **DH/DHE** and **ECDHE/ECDH** rely on different mathematical bases (integer-field vs elliptic-curve).
- **RSA** and **ECDSA** use different signature schemes that will migrate to different PQC replacements.

Grouping based on exact match ensures that each quantum risk level is correctly calculated.

To Provide a Clearer, More Actionable Inventory

In a real environment, endpoints may support dozens of cipher suites, many of which differ only slightly.

- Without grouping, your PQC readiness report would list:
  - 20+ nearly identical lines with minimal variation.
- By grouping logically identical suites:
  - You get one consolidated row per unique combination
    - *DHE (RSA)*
    - *ECDHE (ECDSA)*
    - *ML-KEM (ML-DSA)*
    - *ML-KEM (RSA)*
  - Making it easy to identify which types of handshakes still rely on classical cryptography.

This makes PQC readiness results shorter, cleaner, and directly actionable.

Bound Certificates

What it shows:

- **Certificates bound to services/endpoints**: These are the active digital certificates actually in use by web servers, load balancers, APIs, mail servers, or applications
- **Details captured:**
  - Algorithm type (RSA, ECC, ECDSA, EdDSA)
  - Key length (RSA-2048, RSA-3072, ECC-256, etc.)
  - Issuer (internal CA, public CA, self-signed)
  - Usage (server auth, client auth, code signing, etc.)
- **Not included**: Certificates that simply exist in a trust store (root CAs, unused intermediates) but are not actively bound to a service.

Why it matters:

- **Trust & authentication**
  - Bound certificates are what clients (browsers, apps, APIs) actually see and rely on to establish trust.
  - If these are weak, expired, or misconfigured, secure communication breaks down.
- **Exposure to quantum threats**
  - Most bound certificates today use RSA or ECC. Both will be breakable by a sufficiently powerful quantum computer.
  - Attackers could use harvested traffic and decrypt it later (“store now, decrypt later” risk).
- **Operational risk**
  - Weak key lengths (RSA < 2048, ECC < 256) can already be broken or are considered insufficient.
  - Long-lived certificates increase risk exposure and make migration harder.
- **PQC migration dependency**

- For an organization to adopt PQC or hybrid certificates, visibility into which certificates are actually bound is essential.
- This ensures you can prioritize the right endpoints for testing, replacement, and staged migration.

Action:

- Inventory all bound certificates and document their algorithms and key lengths
- Decommission weak certificates (RSA < 2048, ECC < 256)
- Implement short-lived certificates (≤ 1 year) with automated renewal
- Begin pilots with hybrid/PQC certificates where supported by your CA/vendor
- Align certificate management policies with future NIST PQC standards.

PQC Score Calculator

The PQC Score Calculator for Configuration Scan measures the quantum resilience of system and application cryptographic configurations by analyzing the protocol versions, cipher suites, key exchange mechanisms, and algorithms defined within them.It helps identify whether configurations are quantum-safe, hybrid, or vulnerable to future quantum attacks by assessing the algorithm types, key strengths, and overall cryptographic posture used across the environment

Using standardized criteria based on NIST PQC guidelines and quantum threat models (such as Shor's and Grover's algorithms), the calculator assigns a numerical score that reflects the level of readiness for post-quantum security

Type	Score
Quantum Resistant crypto categories identified	1
Quantum Vulnerable crypto categories identified	0

PQC Score = ∑(Quantum-Resistant Crypto Categories identified) / (Total Crypto Categories) X *10

Explanation

- **∑(Count of Quantum Resistant crypto categories identified)** → The sum of all crypto assets that are identified as Quantum Resistant.
- **Total Crypto categories** → The total number of crypto assets identified across the configurations.
- The result is multiplied by **10** to convert it into a **1–10 scale**.

Eg: Assume you have 100 Crypto assets identified.

Quantum-Vulnerable & Quantum-Resistant values derived from Quantum Readiness column value.

Crypto categories	Total	Total Quantum-Vulnerable	Total Quantum-Resistant
Protocols	50	45	5
Certificates	10	5	5
Cryptographic Library	40	30	10

- Total crypto categories (Protocols+Certificates +Cryptographic library)→ 100
- Total Quantum vulnerable crypto categories identified (Protocols+Certificates +Cryptographic library)→ 80
- Total Quantum-Resistant crypto categories identified(Protocols+Certificates +Cryptographic library) → 20

Example calculation

PQC Score = (20/100)*10 = 2

Threat Level Interpretation

Severity Calculation

Scenario: Apache Application Cipher Discovery

During the PQC readiness scan, the Apache web server exposes the following TLS cipher suites in its configuration.

Cipher Suite	Key Exchange (KX)	Authentication (AUTH)	Encryption (ENC)	Hash (HASH)
TLS_AES_256_GCM_SHA384	ML-KEM	RSA-2048	AES-256	SHA-384
TLS_AES_128_GCM_SHA256	ML-KEM	RSA-2048	AES-128	SHA-256
TLS_ECDH_RSA_AES256_SHA384	ECDH X25519	RSA-2048	AES-256	SHA-384
TLS_CHACHA20_POLY1305_SHA256	ML-KEM	ECDSA	CHACHA20	SHA-256
TLS_AES_256_GCM_SHA384	ML-KEM	ML-DSA	AES-256	SHA-384

These represent the mix of classical and post-quantum (PQC) algorithms configured for HTTPS traffic.

Step 1: Grouping Logic

Cipher suites are grouped by their Key Exchange (KX) and Authentication (AUTH) combinations because these define their quantum risk behavior.

Encryption (ENC) and Hashing (HASH) differences have minimal PQC impact and are averaged per group.

Group	Cipher Suites Included	KX	AUTH
G1	TLS_AES_256_GCM_SHA384 TLS_AES_128_GCM_SHA256	ML-KEM	RSA-2048
G2	TLS_ECDH_RSA_AES256_SHA384	ECDH X25519	RSA-2048
G3	TLS_CHACHA20_POLY1305_SHA256	ML-KEM	ECDSA
G4	TLS_AES_256_GCM_SHA384	ML-KEM	ML-DSA

Step 2: Weighted Scoring Formula

Each function contributes differently to the total score based on impact weight

Function	Weight	Rationale
KX	0.4	Most critical (Harvest-Now, Decrypt-Later risk)
AUTH	0.3	Identity forgery risk
ENC	0.2	Symmetric risk minimal but counted
HASH	0.1	Lowest impact

PQC Weighted Score = (0.40×KX) + (0.30×AUTH) + (0.20×ENC) + (0.10×HASH)

Step 3: Calculate Scores per Cipher Group

Group	KX	AUTH	ENC	HASH	Weighted Score Calculation	Base Score	
G1: (ML-KEM, RSA-2048)	100	0	100	100	(0.4×100) + (0.3×0) + (0.2×100) + (0.1×100) = 70	70	Mo
G2: (X25519, RSA-2048)	0	0	100	100	(0.4×0) + (0.3×0) + (0.2×100) + (0.1×100) = 30	30	Cri

Group	KX	AUTH	ENC	HASH	Weighted Score Calculation	Base Score	
G3: (ML-KEM, ECDSA)	100	0	100	100	(0.4×100) + (0.3×0) + (0.2×100) + (0.1×100) = 70	70	Mo
G4: (ML-KEM, ML-DSA)	100	100	100	100	(0.4×100) + (0.3×100) + (0.2×100) + (0.1×100) = 100	100	Lo

After Applying Gates:

Group	Base Severity	Gate Triggered	Final Severity	Reason
G1	Moderate	AUTH Gate	High	AUTH = RSA-2048 (classical)
G2	Critical	KX Gate	Critical	KX = X25519 (classical)
G3	Moderate	AUTH Gate	High	AUTH = ECDSA (classical)
G4	Low	None	Low	Fully PQC (ML-KEM + ML-DSA)

Step 4: Final Results Summary

Group	Cipher Families	Weighted Score	Final Severity	Migration Priority
G1	ML-KEM + RSA-2048	70	High	Replace RSA with PQC/Hybrid signature (e.g., ML-DSA)
G2	X25519 + RSA-2048	30	Critical	Upgrade both KX and AUTH to PQC
G3	ML-KEM + ECDSA	70	High	Replace ECDSA with PQC signature
G4	ML-KEM + ML-DSA	100	Low	Fully PQC-ready

Range	Severity	Meaning
85–100	Low	Fully PQC-ready
70–84	Moderate	Partial PQC adoption

Range	Severity	Meaning
40–69	High	PQC adoption incomplete
<40	Critical	Classical only — immediate migration required

Note: For Cipher suite gating criteria as mentioned above will be used and severity will not be determined on the actual key exchange and auth algorithms listed below apart from the deprecated algorithms.

If any deprecated algorithm is used in Key exchange or Authentication or Hashing or Encryption ,the protocol severity will be directly marked critical

These are for references to NIST and Shor's analysis on breaking an algorithm:

Key Exchange

Algorithm	NIST / FIPS Status	Quantum Status	PQC Severity
RSA-512	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
RSA-768	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
RSA-1024	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
RSA-1536	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
RSA-2048	Approved (FIPS 186-4)	Quantum Vulnerable	High
RSA-3072	Approved (FIPS 186-4)	Quantum Vulnerable	High
RSA-4096	Approved (FIPS 186-4)	Quantum Vulnerable	High
RSA-6144	Approved (FIPS 186-4)	Quantum Vulnerable	High

Algorithm	NIST / FIPS Status	Quantum Status	PQC Severity
RSA-7680	Not Approved	Quantum Vulnerable	High
RSA-8192	Approved (FIPS 186-4)	Quantum Vulnerable	High
RSA-12288	Not Approved	Quantum Vulnerable	High
RSA-15360	Not Approved	Quantum Vulnerable	High
DH-1024	Approved (SP 800-56A, Deprecated <2048-bit)	Quantum Vulnerable	Critical
DH-2048	Approved (SP 800-56A)	Quantum Vulnerable	High
DH-3072	Approved (SP 800-56A)	Quantum Vulnerable	High
DH-4096	Approved (SP 800-56A)	Quantum Vulnerable	High
DHE-2048	Approved (SP 800-56A)	Quantum Vulnerable	High
DHE-3072	Approved (SP 800-56A)	Quantum Vulnerable	High
DHE-4096	Approved (SP 800-56A)	Quantum Vulnerable	High
ECDH-P192	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
ECDH-P224	Approved (FIPS 186-4, Deprecated)	Quantum Vulnerable	Critical
ECDH-P256	Approved (FIPS 186-4)	Quantum Vulnerable	High
ECDH-P384	Approved (FIPS 186-4)	Quantum Vulnerable	High

Algorithm	NIST / FIPS Status	Quantum Status	PQC Severity
ECDH-P521	Approved (FIPS 186-4)	Quantum Vulnerable	High
ECDH-brainpoolP256r1	Not Approved (RFC 5639, widely used in EU)	Quantum Vulnerable	High
ECDH-brainpoolP384r1	Not Approved (RFC 5639)	Quantum Vulnerable	High
ECDH-brainpoolP512r1	Not Approved (RFC 5639)	Quantum Vulnerable	High
ECDH-secp256k1	Not Approved (Bitcoin/Blockchain curve)	Quantum Vulnerable	High
ECDHE-P192	Approved (FIPS 186-2, Deprecated)	Quantum Vulnerable	Critical
ECDHE-P224	Approved (FIPS 186-4, Deprecated)	Quantum Vulnerable	Critical
ECDHE-P256	Approved (FIPS 186-4)	Quantum Vulnerable	High
ECDHE-P384	Approved (FIPS 186-4)	Quantum Vulnerable	High
ECDHE-P521	Approved (FIPS 186-4)	Quantum Vulnerable	High
ECDHE-brainpoolP256r1	Not Approved (RFC 5639)	Quantum Vulnerable	High
ECDHE-brainpoolP384r1	Not Approved (RFC 5639)	Quantum Vulnerable	High
ECDHE-brainpoolP512r1	Not Approved (RFC 5639)	Quantum Vulnerable	High
ECDHE-secp256k1	Not Approved	Quantum Vulnerable	High
ML-KEM-512	NIST PQC Finalist (FIPS 203 draft)	Quantum Resistant	Low (Quantum Safe)

Algorithm	NIST / FIPS Status	Quantum Status	PQC Severity
ML-KEM-768	NIST PQC Finalist (FIPS 203 draft)	Quantum Resistant	Low (Quantum Safe)
ML-KEM-1024	NIST PQC Finalist (FIPS 203 draft)	Quantum Resistant	Low (Quantum Safe)
SecP256r1MLKEM768	In Draft (NIST migration guidance)	Hybrid	Medium
X25519MLKEM768	In Draft (NIST migration guidance)	Hybrid	Hybrid
SecP384r1MLKEM1024	In Draft (NIST migration guidance)	Hybrid	Hybrid

Authentication

• Certificates rated by risk level

Cert Type	Description	Certificate Validity Window	NIST PQC Timeline	Severity
Classical: RSA ≤ 2048 bits	~112-bit classical strength	After 2030	Insecure beyond 2030	Critical
Classical: RSA ≤ 2048 bits	~112-bit classical strength	On/before 2030	Allowed within window	High
Classical: RSA 2048–3071 bits	~112–128-bit classical strength	After 2035	Insecure beyond 2035	Critical
Classical: RSA 2048–3071 bits	~112–128-bit classical strength	On/before 2035	Allowed within window	High
Classical: RSA ≥ 3072 bits	≥128-bit classical strength	Any	Classical secure	High
Classical: DSA ≤ 1024 bits	Weak / legacy	Any	Already insecure	Critical
Classical: DSA ≤2048 bits	Moderate classical strength	Any	Classical secure	High
Classical: DSA ≥ 3072 bits	Strong classical strength	Any	Classical secure	High

Cert Type	Description	Certificate Validity Window	NIST PQC Timeline	Severity
Classical: ECC < P-256	Weak curve	After 2030	Insecure beyond 2030	Critical
Classical: ECC < P-256	Weak curve	On/before 2030	Allowed within window	High
Classical: ECC ≥ P-256	Modern curve	Any	Classical secure	High
Classical: ECC P-384 / P-521	Strong curve	Any	Classical secure	High
Classical: Ed25519 / Ed448	Modern curve	Any	Classical secure	High
Hybrid: RSA + PQC	Classical + PQC	Any	Recommended now → post-2035	Low (Quantum Safe)
Hybrid: ECC + PQC	Classical + PQC	Any	Recommended now → post-2035	Low (Quantum Safe)
Pure PQC	ML-KEM ML-DSA SLH-DSA	Any	Recommended now → post-2035	Low (Quantum Safe)

Encryption

Algorithm	NIST Status	Severity	PQC Readiness
AES-128	Approved (FIPS 197)	High	Quantum Vulnerable
AES-192	Approved (FIPS 197)	High	Quantum Vulnerable
AES-256	Approved (FIPS 197)	Low (Quantum Safe)	Quantum Resistant

Algorithm	NIST Status	Severity	PQC Readiness
AES-128-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-192-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-256-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-128-CBC	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-192-CBC	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-256-CBC	Approved (FIPS 197 / SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CTR	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-192-CTR	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-256-CTR	Approved (FIPS 197 / SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-GCM	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
AES-192-GCM	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
AES-256-GCM	Approved (FIPS 197 / SP 800-38D)	Low (Quantum Safe)	Quantum Resistant
Camellia-128-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-192-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-256-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	Low (Quantum Safe)	Quantum Resistant

Algorithm	NIST Status	Severity	PQC Readiness
CAMELLIA-128	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-192	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-256	Approved / Recognized (SP 800-57)	Low (Quantum Safe)	Quantum Resistant
Camellia-192-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-256-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	Low (Quantum Safe)	Quantum Resistant
Camellia-128-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
CAMELLIA-128-CTR	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-192-CTR	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-256-CTR	Approved / Recognized (SP 800-57)	Low (Quantum Safe)	Quantum Resistant
CAMELLIA-128-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	High	Quantum Vulnerable
CAMELLIA-192-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	High	Quantum Vulnerable
CAMELLIA-256-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	Low (Quantum Safe)	Quantum Resistant
AES-128-CCM	Approved (FIPS 197 / SP 800-38C)	High	Quantum Vulnerable
AES-192-CCM	Approved (FIPS 197 / SP 800-38C)	High	Quantum Vulnerable
AES-256-CCM	Approved (FIPS 197 / SP 800-38C)	Low (Quantum Safe)	Quantum Resistant

Algorithm	NIST Status	Severity	PQC Readiness
AES-128-OCB	Approved (FIPS 197 / RFC 7253)	High	Quantum Vulnerable
AES-192-OCB	Approved (FIPS 197 / RFC 7253)	High	Quantum Vulnerable
AES-256-OCB	Approved (FIPS 197 / RFC 7253)	Low (Quantum Safe)	Quantum Resistant
AES-128-XTS	Approved (SP 800-38E)	High	Quantum Vulnerable
AES-256-XTS	Approved (SP 800-38E)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB1	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB1	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB1	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB8	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB8	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB8	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-OFB	Approved (SP 800-38A)	High	Quantum Vulnerable

Algorithm	NIST Status	Severity	PQC Readiness
AES-192-OFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-OFB	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-SIV	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-SIV	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-SIV	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
AES-128-WRAP	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-WRAP	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-WRAP	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
AES-128-KW	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-KW	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-KW	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
3DES-112	Approved (FIPS 46-3 / SP 800-67)	High	Quantum Vulnerable
3DES-168	Approved (FIPS 46-3 / SP 800-67)	High	Quantum Vulnerable
Blowfish-128	Not Approved / Legacy	Critical	Quantum Vulnerable
Blowfish-192	Not Approved / Legacy	Critical	Quantum Vulnerable

Algorithm	NIST Status	Severity	PQC Readiness
Blowfish-256	Not Approved / Legacy	Critical	Quantum Vulnerable
Twofish-128	Not Approved / Legacy	High	Quantum Vulnerable
Twofish-192	Not Approved / Legacy	High	Quantum Vulnerable
Twofish-256	Not Approved / Legacy	High	Quantum Vulnerable
Threefish-256	Not Approved / Legacy	High	Quantum Vulnerable
Threefish-512	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
THREEFISH-1024	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
Serpent -128	Not Approved / Legacy	High	Quantum Vulnerable
Serpent -192	Not Approved / Legacy	High	Quantum Vulnerable
Serpent -256	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
ChaCha20	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
XChaCha20	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ChaCha20-Poly1305	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
XChaCha20-Poly1305	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
AES-128-POLY1305	Approved / FIPS 197, SP 800-38D	High	Quantum Vulnerable

Algorithm	NIST Status	Severity	PQC Readiness
AES-192-POLY1305	Approved / FIPS 197, SP 800-38D	High	Quantum Vulnerable
AES-256-POLY1305	Approved / FIPS 197, SP 800-38D	Low (Quantum Safe)	Quantum Resistant
DES-56	Deprecated (FIPS 46-3)	Critical	Quantum Vulnerable
RC2-40	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC2-64	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC2-128	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC4-40	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC2-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-40	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-64	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-32-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-32-192	Deprecated / Weak	Critical	Quantum Vulnerable

Algorithm	NIST Status	Severity	PQC Readiness
RC5-32-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-192	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-192	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC6-128	Not FIPS-approved	High	Quantum Vulnerable
RC6-192	Not FIPS-approved	High	Quantum Vulnerable
RC6-256	Not FIPS-approved	High	Quantum Vulnerable
RC6-512	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
RC6-1024	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
Rijndael-128	Not FIPS-approved	High	Quantum Vulnerable
Rijndael-192	Not FIPS-approved	High	Quantum Vulnerable
Rijndael-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant

Algorithm	NIST Status	Severity	PQC Readiness
IDEA-128	Not NIST-approved / Obsolete	Critical	Quantum Vulnerable
ARIA-128	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CBC	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CBC	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-CBC	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CTR	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CTR	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-CTR	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-GCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-GCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-GCM	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CCM	Not FIPS-approved	High	Quantum Vulnerable

Algorithm	NIST Status	Severity	PQC Readiness
ARIA-256-CCM	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
RC6-128	Not FIPS-approved	Critical	Quantum Vulnerable
RC6-192	Not FIPS-approved	High	Quantum Vulnerable
RC6-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant

Hashing

Algorithm / Variant	NIST/FIPS Status (FIPS/SP)	PQC Severity	PQC Readiness
MD5	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
MD4	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
MD2	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
SHA-0	Deprecated / Legacy	Critical	Quantum Vulnerable
SHA-1	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
SHA-224	Approved (FIPS 180-4)	High	Quantum Vulnerable
SHA-256	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA-384	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA-512	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant

Algorithm / Variant	NIST/FIPS Status (FIPS/SP)	PQC Severity	PQC Readiness
SHA-512/224	Approved (FIPS 180-4)	High	Quantum Vulnerable
SHA-512/256	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA3-224	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-256	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-384	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-512	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
RIPEMD-128	Not FIPS-approved	Critical	Quantum Vulnerable
RIPEMD-160	Not FIPS-approved	Critical	Quantum Vulnerable
RIPEMD-256	Not FIPS-approved	High	Quantum Vulnerable
RIPEMD-320	Not FIPS-approved	High	Quantum Vulnerable
Whirlpool	Not FIPS-approved	High	Quantum Vulnerable
Tiger	Not FIPS-approved	High	Quantum Vulnerable
BLAKE-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
BLAKE-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable

Algorithm / Variant	NIST/FIPS Status (FIPS/SP)	PQC Severity	PQC Readiness
BLAKE-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
BLAKE-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
BLAKE2s-128	Standardized (RFC 7693, not NIST-approved)	Critical	Quantum Vulnerable
BLAKE2s-160	Standardized (RFC 7693, not NIST-approved)	Critical	Quantum Vulnerable
BLAKE2s-224	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2s-256	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-224	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-256	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-384	Standardized (RFC 7693, not NIST-approved)	Low (Quantum Safe)	Quantum Resistant
BLAKE2b-512	Standardized (RFC 7693, not NIST-approved)	Low (Quantum Safe)	Quantum Resistant
BLAKE3	Not Approved (IETF draft, not standardized by NIST)	High	Quantum Vulnerable
KMAC128	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
KMAC256	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
Skein-256-128	Not Approved (SHA-3 competition finalist, not standardized)	Critical	Quantum Vulnerable

Algorithm / Variant	NIST/FIPS Status (FIPS/SP)	PQC Severity	PQC Readiness
Skein-256-160	Not Approved (SHA-3 competition finalist, not standardized)	Critical	Quantum Vulnerable
Skein-256-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-256-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-512-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-1024	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant

Hashing Protocols

Protocol	NIST / FIPS Status	Quantum Status	PQC Severity
TLS 1.0	Not Approved (deprecated, RFC 2246)	Quantum Vulnerable	Critical
TLS 1.1	Not Approved (deprecated, RFC 4346)	Quantum Vulnerable	Critical
SSH 1.0	Not Approved, obsolete	Quantum Vulnerable	Critical
SSL 2.0	Not Approved (deprecated, RFC 6176)	Quantum Vulnerable	Critical
SSL 3.0	Not Approved (deprecated, RFC 6101)	Quantum Vulnerable	Critical

Protocol	NIST / FIPS Status	Quantum Status	PQC Severity
TLS 1.2	Approved	Depend on cipher suite strength	
TLS 1.3	Approved	Depend on cipher suite strength	

Note: Protocol severity is based on key exchange and authentication because those are the parts quantum computers will break first.

Service Binding Information

The Service Binding field captures how a detected service is bound to the network.

It identifies the IP address, port, and (if applicable) the hostname that define where and how the service is listening for incoming connections.

Component	Description
Binding IP Address	The network interface (IPv4 or IPv6) on which the service listens.
Port Number	The specific communication port assigned to the service.
Hostname / Server Name (optional)	The logical name or virtual host defined in the service configuration.

Binding	Meaning
127.0.0.1	Service listens only on the same host (local).
0.0.0.0	Service listens on all available IPv4 interfaces (wildcard).
[::]	Service listens on all available IPv6 interfaces (wildcard).
Specific IP (e.g., 172.16.7.1)	Service listens only on that network interface.

If a service is configured to listen on both IPv4 and IPv6 interfaces simultaneously (for example, 0.0.0.0:443 and [::]:443), the platform denotes this condition with an asterisk (*) in the Service Binding field.

This indicates that the service is dual-stack listening on both IPv4 and IPv6 addresses,

- Hostnames are extracted from service configurations (e.g., server_name, ServerName, or VirtualHost).
- If separate hostnames are defined per address family, both bindings are recorded individually.

Dashboards & Migration Guidance

As part of the **Configuration PQC Readiness Report**, the platform provides dashboards and guided actions to help you understand your cryptographic posture and plan your migration toward post-quantum cryptography.

Dashboards

What it shows:

- **PQC Score:** An overall readiness score (0–100) calculated based on protocols, libraries, ciphers, and certificates
- **Total Endpoints Detected:** All servers, APIs, and devices with cryptographic configurations
- **Total Applications Detected:** Applications and services discovered
- **Quantum Readiness by Library:** Percentage of libraries that are PQC-ready, classical-only, or outdated
- **Quantum Readiness by Protocol:** Breakdown of TLS/SSL protocol usage across environments
- **Risk Levels:** Distribution of Critical, High, Medium, Low, and Unknown crypto risks
- **Key Exchange & Authentication Readiness:** Current use of RSA, ECDHE, hybrid, or PQC-based mechanisms.

Why it matters:

Dashboards provide a centralized view of cryptographic assets across the enterprise. They highlight legacy exposure, quantify PQC readiness, and help prioritize which applications or endpoints should be addressed first. Without visibility at this level, PQC migration planning is fragmented and incomplete.

Migration Guidance

What it shows:

Recommended steps to transition configurations and applications toward PQC readiness

- Retire legacy protocols (SSLv2/3, TLS 1.0/1.1) and enforce TLS 1.3.
- Replace weak or quantum-vulnerable ciphers with strong classical and prepare for hybrid
- Upgrade cryptographic libraries to PQC-supported versions
- Re-issue bound certificates with stronger keys, and plan for hybrid/PQC certificates
- Eliminate RSA/ECC static key exchanges and adopt ephemeral or hybrid key exchanges.

Why it matters:

- Ensures long-term security of sensitive data against “store now, decrypt later” attacks
- Reduces risk of sudden breakage or compliance failure when PQC becomes mandatory
- Provides a clear roadmap from today’s classical cryptography to quantum-resistant standards.

Actions:

- Use the PQC Score as a baseline to set targets for improvement
- Prioritize High-risk items (legacy protocols, weak certificates) first, as they most impact scoring
- Recalculate PQC Score regularly to track progress and demonstrate readiness improvements over time.

For Code Scan Outcome Analysis

This guide outlines the sections of the Code PQC Readiness Report and provides guidance on interpreting the results. It highlights quantum-vulnerable cryptographic algorithms, embedded certificates, and third-party libraries that may impact your organization's migration to post-quantum cryptography.

Scope of Discovery

What it shows:

- Scanning covers:
  - **Source code repositories** (Java, Python, C/C++)
  - **Build artifacts** (JARs, DLLs, binaries)
  - **Embedded certificates** inside applications
  - **Third-party crypto libraries** (OpenSSL, BouncyCastle, WolfSSL, etc.)
  - **Custom libraries** uploaded for scanning.

Why it matters:

- Ensures PQC readiness evaluation includes both application code and dependencies.

Action:

- Ensure all critical repos and binaries are included in the scan scope.

Algorithm Detection

What it shows:

Algorithms used in code are categorized into:

- **Symmetric**
- **Asymmetric**
- **Hash**
- **Key Derivation Functions (KDFs)**
- **Message Authentication Codes**.

Why it matters:

- **RSA/ECC/DH** are quantum-vulnerable.
- **MD5/SHA-1** are already insecure.
- PQC adoption requires **NIST-recommended replacements**.

Action:

- Replace vulnerable algorithms with PQC or hybrid alternatives.
- Ensure KDFs and symmetric algorithms meet NIST security levels.

PQC Score Calculator

The PQC Score Calculator for Code Scan measures the quantum resilience of application source code by analyzing the cryptographic algorithms, key usages, and libraries implemented within it. It helps identify whether the code uses quantum-safe, hybrid, or vulnerable cryptographic components by evaluating the algorithm types, library dependencies, and overall crypto implementation strength

Using standardized criteria based on NIST PQC guidelines and quantum threat models (such as Shor's and Grover's algorithms), the calculator assigns a numerical score that reflects the level of readiness for post-quantum security

Type	Score
Quantum Resistant crypto categories identified	1
Quantum Vulnerable crypto categories identified	0

PQC Score = ∑(Quantum-Resistant Crypto Categories identified) / (Total Crypto Categories) X *10

Explanation

- **∑(Count of Quantum Resistant crypto categories identified)** → The sum of all crypto assets that are identified as Quantum Resistant.
- **Total Crypto categories** → The total number of crypto assets identified across the configurations.
- The result is multiplied by **10** to convert it into a **1–10 scale**

Eg: Assume you have 100 Crypto Category

Quantum-Vulnerable & Quantum-Resistant values derived from Quantum Readiness column value

Tab Name	Crypto categories	Total	Total Quantum-Vulnerable	Total Quantum-Resistant
Direct Cryptographic Usage	Algorithm	50	40	10
Direct Cryptographic Usage	Certificates	10	5	5
Cryptographic Dependancies	Cryptographic Library	40	30	10

- Total crypto categories (Algorithm +Certificates +Cryptographic library)→ 100
- Total Quantum vulnerable crypto categories identified (Algorithm +Certificates +Cryptographic library)→ 75
- Total Quantum-Resistant crypto categories identified (Algorithm +Certificates +Cryptographic library) → 25

Example calculation:

PQC Score = (25/100)*10 = 2.5

Threat Level Interpretation

Severity Assessment (NIST-Aligned)

What it shows:

Algorithms are rated by PQC risk:

- Critical
- High
- Medium
- Low

Note:

- If the cryptographic algorithm is referenced from a constant, variable, configuration, or resolved at runtime and cannot be determined through static analysis, the algorithm cannot be determined and will be marked as Unknown
- If there are no key lengths mentioned in the cryptographic class ,then the severity would be marked as High.

Symmetric Encryption

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
AES-128	Approved (FIPS 197)	High	Quantum Vulnerable
AES-192	Approved (FIPS 197)	High	Quantum Vulnerable
AES-256	Approved (FIPS 197)	Low (Quantum Safe)	Quantum Resistant
AES-128-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-192-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-256-ECB	Approved (FIPS 197)	High	Quantum Vulnerable
AES-128-CBC	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
AES-192-CBC	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-256-CBC	Approved (FIPS 197 / SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CTR	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-192-CTR	Approved (FIPS 197 / SP 800-38A)	High	Quantum Vulnerable
AES-256-CTR	Approved (FIPS 197 / SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-GCM	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
AES-192-GCM	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
AES-256-GCM	Approved (FIPS 197 / SP 800-38D)	Low (Quantum Safe)	Quantum Resistant
Camellia-128-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-192-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-256-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	Low (Quantum Safe)	Quantum Resistant
CAMELLIA-128	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-192	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-256	Approved / Recognized (SP 800-57)	Low (Quantum Safe)	Quantum Resistant

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
Camellia-192-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
Camellia-256-CBC	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	Low (Quantum Safe)	Quantum Resistant
Camellia-128-GCM	Approved (ISO/IEC 18033-3 / JIS X 6319-3)	High	Quantum Vulnerable
CAMELLIA-128-CTR	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-192-CTR	Approved / Recognized (SP 800-57)	High	Quantum Vulnerable
CAMELLIA-256-CTR	Approved / Recognized (SP 800-57)	Low (Quantum Safe)	Quantum Resistant
CAMELLIA-128-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	High	Quantum Vulnerable
CAMELLIA-192-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	High	Quantum Vulnerable
CAMELLIA-256-CCM	Approved / Recognized (SP 800-38C, ISO/IEC 18033-3)	Low (Quantum Safe)	Quantum Resistant
AES-128-CCM	Approved (FIPS 197 / SP 800-38C)	High	Quantum Vulnerable
AES-192-CCM	Approved (FIPS 197 / SP 800-38C)	High	Quantum Vulnerable
AES-256-CCM	Approved (FIPS 197 / SP 800-38C)	Low (Quantum Safe)	Quantum Resistant
AES-128-OCB	Approved (FIPS 197 / RFC 7253)	High	Quantum Vulnerable
AES-192-OCB	Approved (FIPS 197 / RFC 7253)	High	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
AES-256-OCB	Approved (FIPS 197 / RFC 7253)	Low (Quantum Safe)	Quantum Resistant
AES-128-XTS	Approved (SP 800-38E)	High	Quantum Vulnerable
AES-256-XTS	Approved (SP 800-38E)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB1	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB1	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB1	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-CFB8	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-CFB8	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-256-CFB8	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-OFB	Approved (SP 800-38A)	High	Quantum Vulnerable
AES-192-OFB	Approved (SP 800-38A)	High	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
AES-256-OFB	Approved (SP 800-38A)	Low (Quantum Safe)	Quantum Resistant
AES-128-SIV	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-SIV	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-SIV	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
AES-128-WRAP	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-WRAP	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-WRAP	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
AES-128-KW	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-192-KW	Approved (SP 800-38F)	High	Quantum Vulnerable
AES-256-KW	Approved (SP 800-38F)	Low (Quantum Safe)	Quantum Resistant
3DES-112	Approved (FIPS 46-3 / SP 800-67)	High	Quantum Vulnerable
3DES-168	Approved (FIPS 46-3 / SP 800-67)	High	Quantum Vulnerable
Blowfish-128	Not Approved / Legacy	Critical	Quantum Vulnerable
Blowfish-192	Not Approved / Legacy	Critical	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
Blowfish-256	Not Approved / Legacy	Critical	Quantum Vulnerable
Twofish-128	Not Approved / Legacy	High	Quantum Vulnerable
Twofish-192	Not Approved / Legacy	High	Quantum Vulnerable
Twofish-256	Not Approved / Legacy	High	Quantum Vulnerable
Threefish-256	Not Approved / Legacy	High	Quantum Vulnerable
Threefish-512	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
THREEFISH-1024	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
Serpent -128	Not Approved / Legacy	High	Quantum Vulnerable
Serpent -192	Not Approved / Legacy	High	Quantum Vulnerable
Serpent -256	Not Approved / Legacy	Low (Quantum Safe)	Quantum Resistant
ChaCha20	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
XChaCha20	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ChaCha20-Poly1305	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
XChaCha20-Poly1305	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
AES-128-POLY1305	Approved / FIPS 197, SP 800-38D	High	Quantum Vulnerable
AES-192-POLY1305	Approved / FIPS 197, SP 800-38D	High	Quantum Vulnerable
AES-256-POLY1305	Approved / FIPS 197, SP 800-38D	Low (Quantum Safe)	Quantum Resistant
DES-56	Deprecated (FIPS 46-3)	Critical	Quantum Vulnerable
RC2-40	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC2-64	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC2-128	Deprecated (SP 800-38 / Legacy)	Critical	Quantum Vulnerable
RC4-40	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC2-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-40	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-64	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC4-256	Deprecated / Weak	Critical	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
RC5-32-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-32-192	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-32-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-192	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-64-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-128	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-192	Deprecated / Weak	Critical	Quantum Vulnerable
RC5-128-256	Deprecated / Weak	Critical	Quantum Vulnerable
RC6-128	Not FIPS-approved	High	Quantum Vulnerable
RC6-192	Not FIPS-approved	High	Quantum Vulnerable
RC6-256	Not FIPS-approved	High	Quantum Vulnerable
RC6-512	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
RC6-1024	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
Rijndael-128	Not FIPS-approved	High	Quantum Vulnerable
Rijndael-192	Not FIPS-approved	High	Quantum Vulnerable
Rijndael-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
IDEA-128	Not NIST-approved / Obsolete	Critical	Quantum Vulnerable
ARIA-128	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CBC	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CBC	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-CBC	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CTR	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CTR	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-CTR	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-GCM	Not FIPS-approved	High	Quantum Vulnerable

Symmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC Readiness
ARIA-192-GCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-GCM	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
ARIA-128-CCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-192-CCM	Not FIPS-approved	High	Quantum Vulnerable
ARIA-256-CCM	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant
RC6-128	Not FIPS-approved	Critical	Quantum Vulnerable
RC6-192	Not FIPS-approved	High	Quantum Vulnerable
RC6-256	Not FIPS-approved	Low (Quantum Safe)	Quantum Resistant

Asymmetric Encryption

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
RSA-512	Approved (Deprecated, FIPS 186-4 / SP 800-131A)	Critical	Quantum Vulnerable
RSA-768	Approved (Deprecated, FIPS 186-4 / SP 800-131A)	Critical	Quantum Vulnerable
RSA-1024	Approved (Deprecated, FIPS 186-4 / SP 800-131A)	Critical	Quantum Vulnerable
RSA-1536	Approved (Deprecated, FIPS 186-4 / SP 800-131A)	Critical	Quantum Vulnerable

Asymmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
RSA-2048	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-3072	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-4096	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-6144	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-7680	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-8192	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-12288	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-15360	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-PSS-2048	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-PSS-3072	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-PSS-4096	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-PSS-7680	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
RSA-PSS-15360	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
DSA-1024	Approved (Deprecated, FIPS 186-3 / 186-4 / SP 800-131A)	Critical	Quantum Vulnerable

Asymmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
DSA-2048	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
DSA-3072	Approved (FIPS 186-4 / SP 800-131A)	High	Quantum Vulnerable
DSA-4096	Approved (FIPS 186-4)	High	Quantum Vulnerable
DSA-8192	Approved (FIPS 186-4)	High	Quantum Vulnerable
DH-1024	Approved (Deprecated, FIPS 186-4 / SP 800-56A Rev. 3)	Critical	Quantum Vulnerable
DH-2048	Approved (FIPS 186-4 / SP 800-56A Rev. 3)	High	Quantum Vulnerable
DH-3072	Approved (FIPS 186-4 / SP 800-56A Rev. 3)	High	Quantum Vulnerable
DHE-2048	Approved (FIPS 186-4 / SP 800-56A Rev. 3)	High	Quantum Vulnerable
DHE-3072	Approved (FIPS 186-4 / SP 800-56A Rev. 3)	High	Quantum Vulnerable
DHE-4096	Approved (FIPS 186-4 / SP 800-56A Rev. 3)	High	Quantum Vulnerable
ECDSA-P192	Approved (Deprecated, FIPS 186-4 / SP 800-186)	Critical	Quantum Vulnerable
ECDSA-P224	Approved (FIPS 186-4 / SP 800-186)	High	Quantum Vulnerable
ECDSA-P256	Approved (FIPS 186-4 / SP 800-186)	High	Quantum Vulnerable
ECDSA-P384	Approved (FIPS 186-4)	High	Quantum Vulnerable

Asymmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
ECDSA-P521	Approved (FIPS 186-4)	High	Quantum Vulnerable
ECDSA-brainpoolP256r1	Not Approved	High	Quantum Vulnerable
ECDSA-brainpoolP384r1	Not Approved	High	Quantum Vulnerable
ECDSA-brainpoolP512r1	Not Approved	High	Quantum Vulnerable
ECDSA-secp256k1	Not Approved	High	Quantum Vulnerable
ECDH-P192	Approved (Deprecated, FIPS 186-4)	Critical	Quantum Vulnerable
ECDH-P224	Approved (FIPS 186-4)	High	Quantum Vulnerable
ECDH-P256	Approved (FIPS 186-4)	High	Quantum Vulnerable
ECDH-P384	Approved (FIPS 186-4)	High	Quantum Vulnerable
ECDH-P521	Approved (FIPS 186-4)	High	Quantum Vulnerable
ECDH-brainpoolP256r1	Not Approved	High	Quantum Vulnerable
ECDH-brainpoolP384r1	Not Approved	High	Quantum Vulnerable
ECDH-brainpoolP512r1	Not Approved	High	Quantum Vulnerable
ECDH-secp256k1	Not Approved	High	Quantum Vulnerable

Asymmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
Ed25519	Not Approved (draft RFC / NIST not yet standardized)	High	Quantum Vulnerable
Ed448	Not Approved (draft RFC / NIST not yet standardized)	High	Quantum Vulnerable
X25519	Not Approved (draft RFC / NIST not yet standardized)	High	Quantum Vulnerable
X448	Not Approved (draft RFC / NIST not yet standardized)	High	Quantum Vulnerable
ML-DSA-44	FIPS 204	Low (Quantum Safe)	Quantum Resistant
ML-DSA-65	FIPS 204	Low (Quantum Safe)	Quantum Resistant
ML-DSA-87	FIPS 204	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-128s	FIPS 205	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-128f	FIPS 205	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-192s	FIPS 205	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-192f	FIPS 205	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-256s	FIPS 205	Low (Quantum Safe)	Quantum Resistant
SLH-DSA-SHA2-256f	FIPS 205	Low (Quantum Safe)	Quantum Resistant
FALCON-512	Draft FIPS 206	Low (Quantum Safe)	Quantum Resistant

Asymmetric Encryption (continued)

Algorithm	NIST Status	Severity based on Grovers/Shor Analysis	PQC readiness
FALCON-1024	Draft FIPS 206	Low (Quantum Safe)	Quantum Resistant

Hash Function

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
MD5	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
MD4	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
MD2	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
SHA-0	Deprecated / Legacy	Critical	Quantum Vulnerable
SHA-1	Deprecated / Legacy (FIPS 180-4)	Critical	Quantum Vulnerable
SHA-224	Approved (FIPS 180-4)	High	Quantum Vulnerable
SHA-256	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA-384	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA-512	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant
SHA-512/224	Approved (FIPS 180-4)	High	Quantum Vulnerable
SHA-512/256	Approved (FIPS 180-4)	Low (Quantum Safe)	Quantum Resistant

Hash Function (continued)

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
SHA3-224	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-256	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-384	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
SHA3-512	Approved (FIPS 202 / SP 800-185)	Low (Quantum Safe)	Quantum Resistant
RIPEMD-128	Not FIPS-approved	Critical	Quantum Vulnerable
RIPEMD-160	Not FIPS-approved	Critical	Quantum Vulnerable
RIPEMD-256	Not FIPS-approved	High	Quantum Vulnerable
RIPEMD-320	Not FIPS-approved	High	Quantum Vulnerable
Whirlpool	Not FIPS-approved	High	Quantum Vulnerable
Tiger	Not FIPS-approved	High	Quantum Vulnerable
BLAKE-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
BLAKE-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
BLAKE-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
BLAKE-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant

Hash Function (continued)

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
BLAKE2s-128	Standardized (RFC 7693, not NIST-approved)	Critical	Quantum Vulnerable
BLAKE2s-160	Standardized (RFC 7693, not NIST-approved)	Critical	Quantum Vulnerable
BLAKE2s-224	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2s-256	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-224	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-256	Standardized (RFC 7693, not NIST-approved)	High	Quantum Vulnerable
BLAKE2b-384	Standardized (RFC 7693, not NIST-approved)	Low (Quantum Safe)	Quantum Resistant
BLAKE2b-512	Standardized (RFC 7693, not NIST-approved)	Low (Quantum Safe)	Quantum Resistant
BLAKE3	Not Approved (IETF draft, not standardized by NIST)	High	Quantum Vulnerable
KMAC128	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
KMAC256	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
Skein-256-128	Not Approved (SHA-3 competition finalist, not standardized)	Critical	Quantum Vulnerable
Skein-256-160	Not Approved (SHA-3 competition finalist, not standardized)	Critical	Quantum Vulnerable
Skein-256-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable

Hash Function (continued)

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
Skein-256-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-224	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-256	Not Approved (SHA-3 competition finalist, not standardized)	High	Quantum Vulnerable
Skein-512-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-512-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-384	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-512	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant
Skein-1024-1024	Not Approved (SHA-3 competition finalist, not standardized)	Low (Quantum Safe)	Quantum Resistant

Message Authentication Codes

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
HMAC-SHA1	Approved (FIPS 198-1)	Critical	Quantum Vulnerable
HMAC-SHA224	Approved (FIPS 198-1)	High	Quantum Vulnerable
HMAC-SHA256	Approved (FIPS 198-1)	Low (Quantum Safe)	Quantum Resistant
HMAC-SHA384	Approved (FIPS 198-1)	Low (Quantum Safe)	Quantum Resistant

Message Authentication Codes (continued)

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
HMAC-SHA512	Approved (FIPS 198-1)	Low (Quantum Safe)	Quantum Resistant
CMAC-AES-128	Approved (FIPS 197 / SP 800-38B)	High	Quantum Vulnerable
CMAC-AES-192	Approved (FIPS 197 / SP 800-38B)	High	Quantum Vulnerable
CMAC-AES-256	Approved (FIPS 197 / SP 800-38B)	Low (Quantum Safe)	Quantum Resistant
GMAC-AES-128	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
GMAC-AES-192	Approved (FIPS 197 / SP 800-38D)	High	Quantum Vulnerable
GMAC-AES-256	Approved (FIPS 197 / SP 800-38D)	Low (Quantum Safe)	Quantum Resistant
Poly1305-128 (ChaCha20-Poly1305)	Not explicitly FIPS-approved	Low (Quantum Safe)	Quantum Resistant
Poly1305-XChaCha20-128	Not explicitly FIPS-approved	Low (Quantum Safe)	Quantum Resistant
KMAC128	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
KMAC256	Approved (SP 800-185)	Low (Quantum Safe)	Quantum Resistant
DES-CBC-MAC-56	Deprecated / Legacy (FIPS 46-3)	Critical	Quantum Vulnerable
3DES-CBC-MAC-112	Deprecated / Legacy (FIPS 46-3)	Critical	Quantum Vulnerable
3DES-CBC-MAC-168	Deprecated / Legacy (FIPS 46-3)	Critical	Quantum Vulnerable

Message Authentication Codes (continued)

Algorithm	NIST/FIPS Status (FIPS/SP)	Severity based on Grovers/Shor Analysis	Quantum Readiness
HMAC-MD5	Deprecated / Legacy	Critical	Quantum Vulnerable
MD5-CBC-MAC-56	Deprecated / Legacy	Critical	Quantum Vulnerable
MD5-CBC-MAC-128	Deprecated / Legacy	Critical	Quantum Vulnerable

Key Derivation Function

Algorithm	NIST Status	Severity based on Grovers/ Shor Analysis	Quantum Readiness
PBKDF1-HMAC-MD5	Deprecated / Legacy	Critical	Quantum Vulnerable
PBKDF1-HMAC-SHA1	Deprecated / Legacy (FIPS 180-1 / PKCS #5)	Critical	Quantum Vulnerable
PBKDF2-HMAC-MD5	Deprecated / Legacy	Critical	Quantum Vulnerable
PBKDF2-HMAC-SHA1	Approved (PKCS #5 / SP 800-132)	Critical	Quantum Vulnerable
PBKDF2-HMAC-SHA224	Approved (PKCS #5 / SP 800-132)	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
PBKDF2-HMAC-SHA256	Approved (PKCS #5 / SP 800-132)	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
PBKDF2-HMAC-SHA384	Approved (PKCS #5 / SP 800-132)	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
PBKDF2-HMAC-SHA512	Approved (PKCS #5 / SP 800-132)	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
HKDF-HMAC-SHA1	Not explicitly FIPS-approved, widely used	Critical	Quantum Vulnerable

Key Derivation Function (continued)

Algorithm	NIST Status	Severity based on Grovers/ Shor Analysis	Quantum Readiness
HKDF-HMAC-SHA224	Not explicitly FIPS-approved, widely used	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
HKDF-HMAC-SHA256	Not explicitly FIPS-approved, widely used	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
HKDF-HMAC-SHA384	Not explicitly FIPS-approved, widely used	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
HKDF-HMAC-SHA512	Not explicitly FIPS-approved, widely used	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
Argon2d	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
Argon2i	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
Argon2id	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
bcrypt-HMAC-SHA1	Not FIPS-approved	High	Quantum Vulnerable
bcrypt-HMAC-SHA224	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
bcrypt-HMAC-SHA256	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
bcrypt-HMAC-SHA384	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
bcrypt-HMAC-SHA512	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
scrypt-HMAC-SHA1	Not FIPS-approved	High	Quantum Vulnerable
scrypt-HMAC-SHA224	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant

Key Derivation Function (continued)

Algorithm	NIST Status	Severity based on Grovers/ Shor Analysis	Quantum Readiness
scrypt-HMAC-SHA256	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
scrypt-HMAC-SHA384	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant
scrypt-HMAC-SHA512	Not FIPS-approved	Low (Quantum Safe) (Quantum Safe)	Quantum Resistant

Why it matters:

• Provides a prioritized list of risks for remediation.

Action:

• Focus on replacing Critical and High algorithms first.

Certificates in Code

What it shows:

• Discovery of **hardcoded or embedded certificates** in application code, configs, or keystores.
• Classification: internal vs public-facing certificates.
• PQC readiness of certificate algorithms and signature schemes.

Why it matters:

• Hardcoded certificates are often forgotten but may secure APIs or internal services.
• These certs must follow the same PQC migration plan as server certs.

Action:

• Replace vulnerable embedded certs with PQC-ready or hybrid certs.

Crypto Libraries

What it shows:

- Detection of **standard crypto libraries** (e.g., OpenSSL, BouncyCastle, WolfSSL).
- Detection of **custom libraries** uploaded by the user for PQC evaluation.
- Library PQC readiness (support for hybrid or PQC algorithms).

Why it matters:

- Applications inherit the cryptographic strength of their libraries.
- Non-standard/custom libraries may hide risks if not reviewed.

Action:

- Upgrade libraries to **PQC-ready versions**.
- Submit custom/internal libraries for analysis to ensure coverage.

Quantum Readiness Classification

What it shows:

Algorithms, certificates, and libraries are classified as:

- **Classical** → Quantum-vulnerable
- **Hybrid** → Transitional (RSA/ECC + PQC)
- **PQC-only** → Quantum-safe

Why it matters:

- Gives a single **maturity snapshot** of application crypto posture.

Action:

- Track transition progress from **Classical** → **Hybrid** → **PQC only**.

High-Risk Findings

What it shows:

- Algorithms or certs using **deprecated schemes** (MD5, SHA-1, DES, 3DES, weak RSA/ECC).
- Libraries that are outdated or lack PQC support.

Why it matters:

- High-risk findings can lead to **application-level compromise** or **regulatory non-compliance**.

Action:

- Prioritize replacing or upgrading high-risk crypto usage.

Dashboards

What it shows:

- **PQC Readiness Score:** % PQC-ready vs vulnerable algorithms/libraries
- **Readiness by Language:** Crypto usage breakdown by programming language
- **Readiness by Repository:** PQC maturity per repo/project
- **Library Readiness:** Standard and custom libraries by PQC support level
- **Algorithm Summary:** Symmetric, asymmetric, hash, KDF usage trends
- **Certificates in Code:** PQC readiness of embedded certificates
- **Quantum Readiness Trend:** Migration progress over time.

Why it matters:

- Provides both developer and leadership visibility into PQC migration status.

Action:

- Use dashboards to track and report migration progress across teams.

Migration Guidance

What it shows:

- Recommended steps to transition to PQC ready code:
  - Replace vulnerable algorithms with NIST-approved PQC or hybrid alternatives.
  - Upgrade crypto libraries to PQC-supported versions.
  - Re-issue embedded certificates with PQC or hybrid certs.
  - Enforce minimum key sizes and secure KDFs.

Why it matters:

- Prevents sudden breakage or compliance failures when PQC adoption becomes mandatory.

Action:

- Follow a phased approach: **Classical** → **Hybrid** → **PQC-only**.