Migration to Post-Quantum Cryptography
Quantum Readiness: Cryptographic Discovery

Volume B:
Approach, Architecture, and Security Characteristics of Public Key Application Discovery Tools

William Newhouse
Murugiah Souppaya
National Institute of Standards and Technology
Rockville, Maryland

William Barker
Dakota Consulting
Silver Spring, Maryland

Chris Brown
The MITRE Corporation
Mclean, Virginia

Panos Kampanakis
Amazon Web Services, Inc. (AWS)
Arlington, Virginia

Marc Manzano
SandboxAQ
Palo Alto, California

David McGrew
Cisco Systems, Inc.
San Jose, California

Anne Dames
IBM
Yorktown Heights, New York

Vladimir Soukharev
InfoSec Global
Toronto, Canada

Philip Lafrance
ISARA Corporation
Ontario, Canada

Anthony Hu
wolfSSL
Seattle, Washington

David Hook
Keyfactor
Independence, Ohio

Raul Garcia
Microsoft
Redmond, Washington

Evgeny Gervis
SafeLogic, Inc.
Palo Alto, California

Eunkyung Kim
Changhoon Lee
Samsung SDS Co., Ltd.
Seoul, Republic of South Korea

December 2023
PRELIMINARY DRAFT

This publication is available free of charge from
DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.


FEEDBACK

This initial draft offers: (1) a functional test plan that exercises the cryptographic discovery tools to determine baseline capabilities; (2) a use case scenario to provide context and scope for our demonstration; (3) an examination of the threats addressed in this demonstration; (4) a multifaceted approach to the discovery process that most organizations can start today; and (5) a high-level architecture based on our use case that integrates contributed discovery tools in our lab.

You can improve this initial public draft by submitting comments. We are always seeking feedback on our publications and how they support our readers’ needs. We are particularly interested in learning from readers if this initial draft is helpful to you and what you want to see covered in future versions of this publication.

The project will soon have a repository in https://github.com/usnistgov to provide the data shared in this initial draft and relevant data created by the project in the future.

Comments on this publication may be submitted to: applied-crypto-pqc@nist.gov.

Public comment period: December 19, 2023 through February 25, 2024.

All comments are subject to release under the Freedom of Information Act.

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov
NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses’ most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in information technology security—the NCCoE applies standards and best practices to develop modular, adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cybersecurity Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Maryland.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov/. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication 1800 series) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align with relevant standards and best practices, and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

KEYWORDS

algorithm; cryptography; encryption; identity management; key establishment and management; post-quantum cryptography; public-key cryptography; quantum-resistant; vulnerable cryptography discovery

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameron Bytheway</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Torben Hansen</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Name</td>
<td>Organization</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Brian Jarvis</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Jake Massimo</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Wesley Rosenblum</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Martin Schaef</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Alex Weibel</td>
<td>Amazon Web Services, Inc. (AWS)</td>
</tr>
<tr>
<td>Avani Wildani</td>
<td>Cloudflare, Inc.</td>
</tr>
<tr>
<td>Natasha Eastman</td>
<td>Cybersecurity and Infrastructure Security Agency (CISA)</td>
</tr>
<tr>
<td>Garfield Jones</td>
<td>Cybersecurity and Infrastructure Security Agency (CISA)</td>
</tr>
<tr>
<td>Nancy Pomerleau</td>
<td>Cybersecurity and Infrastructure Security Agency (CISA)</td>
</tr>
<tr>
<td>Judith Furlong</td>
<td>Dell Technologies</td>
</tr>
<tr>
<td>Corey Bonnell</td>
<td>DigiCert</td>
</tr>
<tr>
<td>Avesta Hojjati</td>
<td>DigiCert</td>
</tr>
<tr>
<td>Boris Balacheff</td>
<td>HP, Inc.</td>
</tr>
<tr>
<td>Tommy Charles</td>
<td>HP, Inc.</td>
</tr>
<tr>
<td>Thalia Laing</td>
<td>HP, Inc.</td>
</tr>
<tr>
<td>Wai Choi</td>
<td>IBM</td>
</tr>
<tr>
<td>Christopher Meyer</td>
<td>IBM</td>
</tr>
<tr>
<td>Michael Osborne</td>
<td>IBM</td>
</tr>
<tr>
<td>Emily Ratliff</td>
<td>IBM</td>
</tr>
<tr>
<td>Kelsey Holler</td>
<td>Information Security Corporation</td>
</tr>
<tr>
<td>Name</td>
<td>Organization</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Philip George</td>
<td>InfoSec Global</td>
</tr>
<tr>
<td>York Lam</td>
<td>InfoSec Global</td>
</tr>
<tr>
<td>Ninaa Persad</td>
<td>InfoSec Global</td>
</tr>
<tr>
<td>Julien Probst</td>
<td>InfoSec Global</td>
</tr>
<tr>
<td>Jackie Parker</td>
<td>ISARA Corporation</td>
</tr>
<tr>
<td>Rob Williams</td>
<td>ISARA Corporation</td>
</tr>
<tr>
<td>Atsushi Yamada</td>
<td>ISARA Corporation</td>
</tr>
<tr>
<td>Hubert Le Van Gong</td>
<td>JPMorgan Chase Bank, N.A.</td>
</tr>
<tr>
<td>John M. Chan</td>
<td>JPMorgan Chase Bank, N.A.</td>
</tr>
<tr>
<td>Roy Basmacier</td>
<td>Keyfactor</td>
</tr>
<tr>
<td>Alexander Scheel</td>
<td>Keyfactor</td>
</tr>
<tr>
<td>Ted Shorter</td>
<td>Keyfactor</td>
</tr>
<tr>
<td>Bryan Uhri</td>
<td>Keyfactor</td>
</tr>
<tr>
<td>Benjamin Rodes</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Lily Chen</td>
<td>National Institute of Standards and Technology (NIST)</td>
</tr>
<tr>
<td>David Cooper</td>
<td>National Institute of Standards and Technology (NIST)</td>
</tr>
<tr>
<td>Daniel Eliot</td>
<td>National Institute of Standards and Technology (NIST)</td>
</tr>
<tr>
<td>Dustin Moody</td>
<td>National Institute of Standards and Technology (NIST)</td>
</tr>
<tr>
<td>Andy Regenscheid</td>
<td>National Institute of Standards and Technology (NIST)</td>
</tr>
<tr>
<td>Mike Jenkins</td>
<td>National Security Agency (NSA)</td>
</tr>
<tr>
<td>Name</td>
<td>Organization</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Brendan Zember</td>
<td>National Security Agency (NSA)</td>
</tr>
<tr>
<td>Sean Morgan</td>
<td>Palo Alto Networks Public Sector, LLC</td>
</tr>
<tr>
<td>Graeme Hickey</td>
<td>PQShield</td>
</tr>
<tr>
<td>Michael Hutter</td>
<td>PQShield</td>
</tr>
<tr>
<td>Kris Kwiatkowski</td>
<td>PQShield</td>
</tr>
<tr>
<td>Ben Packman</td>
<td>PQShield</td>
</tr>
<tr>
<td>Axel Poschmann</td>
<td>PQShield</td>
</tr>
<tr>
<td>Jihoon Cho</td>
<td>Samsung SDS Co., Ltd.</td>
</tr>
<tr>
<td>Yoonchan Jhi</td>
<td>Samsung SDS Co., Ltd.</td>
</tr>
<tr>
<td>Hyojin Yoon</td>
<td>Samsung SDS Co., Ltd.</td>
</tr>
<tr>
<td>Hunhee Yu</td>
<td>Samsung SDS Co., Ltd.</td>
</tr>
<tr>
<td>Eric Ahlstrom</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Jackson Beall</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Nikolai Chernyy</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Nick Hamilton</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Tarun Sibal</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Michael Toney</td>
<td>SandboxAQ</td>
</tr>
<tr>
<td>Daniel Cuthbert</td>
<td>Santander</td>
</tr>
<tr>
<td>Jaime Gomez</td>
<td>Santander</td>
</tr>
<tr>
<td>Robert Burns</td>
<td>Thales DIS CPL USA, Inc.</td>
</tr>
</tbody>
</table>
The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

<table>
<thead>
<tr>
<th>Migration to Post Quantum Cryptography Technology Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>Jane Gilbert</td>
</tr>
<tr>
<td>D'nan Godfrey</td>
</tr>
<tr>
<td>Gina Scinta</td>
</tr>
<tr>
<td>Kaitlyn Laohoo</td>
</tr>
<tr>
<td>Neil McNab</td>
</tr>
<tr>
<td>Jessica Walton</td>
</tr>
<tr>
<td>Volker Krummel</td>
</tr>
<tr>
<td>Lee E. Sattler</td>
</tr>
<tr>
<td><strong>Amazon Web Services, Inc. (AWS)</strong></td>
</tr>
<tr>
<td><strong>Cisco Systems, Inc.</strong></td>
</tr>
<tr>
<td><strong>Cloudflare, Inc.</strong></td>
</tr>
<tr>
<td><strong>Crypto4A Technologies, Inc.</strong></td>
</tr>
<tr>
<td><strong>CryptoNext Security</strong></td>
</tr>
<tr>
<td><strong>Cybersecurity and Infrastructure Security Agency (CISA)</strong></td>
</tr>
<tr>
<td><strong>Dell Technologies</strong></td>
</tr>
</tbody>
</table>
**Migration to Post Quantum Cryptography Technology Collaborators**

<table>
<thead>
<tr>
<th>DigiCert</th>
<th>Palo Alto Networks Public Sector, LLC</th>
<th>Verizon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrust</td>
<td>PQShield</td>
<td>VMware, Inc.</td>
</tr>
<tr>
<td>HP, Inc.</td>
<td>QuantumXchange</td>
<td>wolfSSL</td>
</tr>
<tr>
<td>IBM</td>
<td>SafeLogic, Inc.</td>
<td></td>
</tr>
</tbody>
</table>

**DOCUMENT CONVENTIONS**

The terms “shall” and “shall not” indicate requirements to be followed strictly to conform to the publication and from which no deviation is permitted. The terms “should” and “should not” indicate that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. The terms “may” and “need not” indicate a course of action permissible within the limits of the publication. The terms “can” and “cannot” indicate a possibility and capability, whether material, physical, or causal.

**CALL FOR PATENT CLAIMS**

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

1. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or

2. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.
Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: applied-crypto-pqc@nist.gov.
### Contents

#### 1 Summary

1.1 Challenge ................................................................. 1

1.2 Outcomes ............................................................... 2

1.3 Preparing for the New Post-Quantum Standards ................. 2

1.3.1 Public-Key Cryptographic Technologies .................. 3

1.4 Demonstration Activity ............................................ 5

#### 2 How to Use This Guide ............................................... 5

#### 3 Approach: The Migration to Post-Quantum Cryptography Project’s Contribution to Quantum Readiness ........................................... 6

3.1 Audience .................................................................... 7

3.2 Scope ........................................................................ 7

3.2.1 Example Discovery Scenario for a Medium-Sized Business .... 7

3.2.2 Project Execution of Example Scenario in Our Lab ........ 9

3.2.3 Vulnerable Cryptographic Algorithms ..................... 12

3.3 Terminology ............................................................ 14

3.4 Risk Assessment .......................................................... 14

3.4.1 Threats to Classical Cryptography ......................... 15

3.4.2 Vulnerabilities ...................................................... 16

3.4.3 Survey of Risk Methodologies ............................. 18

#### 4 Architecture ............................................................ 21

4.1 Architecture Description ............................................. 21

4.1.1 Protecting the Code Development Pipeline ................ 21

4.1.2 Operational Systems and Applications ..................... 23

4.1.3 Transport Protocols and Network Services ............... 24

4.1.4 Common Output Elements for Identifying Vulnerable Systems .... 25

#### 5 Technologies ................................................................ 33

5.1 Consortium Members .................................................. 33

5.1.1 Cisco .................................................................. 33

5.1.2 IBM .................................................................... 33

5.1.3 Infosec Global (ISG) ............................................. 34

5.1.4 ISARA Corporation ............................................... 34
Figure 3 Conceptual CI Pipeline .......................................................... 22
Figure 4 Sensor data flow ............................................................... 23
Figure 5 Passive Network Discovery ................................................... 24

List of Tables

Table 1 SSDF Practice Groups .......................................................... 10
Table 2 Security Characteristic Mapping to SSDF Tasks ...................... 10
Table 3 Scope of Vulnerable Algorithms .......................................... 13
Table 4 Submitted CWE .................................................................. 17
Table 5 Software Development Components .................................... 22
Table 6 Network-based discovery data elements .................................. 25
Table 7 tool component property specification ................................... 29
Table 8 runs.tool.rules property specification ................................... 30
Table 9 runs.results property specification ....................................... 31
Table 10 runs.tools.rules.message Elements ..................................... 32
Table 11 Vulnerable Algorithm Identifiers .......................................... 32
Table 12 Products and Technologies .................................................. 36
1 Summary

In 2021, the National Institute of Standards and Technology (NIST) published CSWP-15, Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms [1], and the National Cybersecurity Center of Excellence (NCCoE) initiated the Migration to Post Quantum Cryptography project [2] to develop practices and demonstrate capabilities for easing migration from the current set of public-key cryptographic algorithms to replacement algorithms that are resistant to quantum-computer-based attacks. The replacement algorithms were then identified and selected under a NIST post-quantum standards program initiated in 2016. Subsequently, on May 4, 2022, the White House issued a National Security Memorandum (NSM) on “Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems.” [3] This NSM reinforced the NCCoE’s project priority by directing NIST to establish a Migration to Post-Quantum Cryptography Project “as the United States begins the multi-year process of migrating vulnerable computer systems to quantum-resistant cryptography.”

The advent of a cryptanalytically relevant quantum computer (CRQC) will render current standard public-key cryptographic algorithms ineffective. These public-key algorithms are widely used in protecting the integrity and confidentiality of digital information. Replacement of cryptographic algorithms is both technically and logistically challenging. It can take years or even decades to complete.

Working with the public and private sectors to address cybersecurity challenges posed by the transition to quantum-resistant cryptography, the NCCoE is undertaking a practical demonstration of technology and tools that can assist organizations to develop a migration plan, sometimes called a quantum readiness roadmap. Two demonstration activities were selected to enable and inform migration. One demonstration is focused on tools that discover where and how public key algorithms are being used; the second involves experiments that measure the performance of the emerging quantum-resistant algorithms for security functions and protocols that are currently reliant on quantum-vulnerable algorithms.

This volume covers the demonstration activities focused on the use of automated discovery tools. These tools identify instances of quantum-vulnerable public-key algorithms that are widely deployed across an organization to create a cryptographic algorithm inventory that will help an organization to develop its migration roadmap.

A separate volume will be published on the development and improvement of migration strategies enabled by demonstrating the interoperability and performance of post-quantum algorithm implementations and sharing our tools and findings with standards developing organizations and industry sectors reliant on cryptographic protection.

The reader should note this is a preliminary draft of the document, and while the content is considered to be stable, changes are expected to occur. There are gaps in the content and the overall document is still incomplete. NIST welcomes early informal feedback and comments, which will be adjudicated after the specified public comment period. Organizations may consider experimenting with guidelines, with the understanding that they will identify gaps and challenges.
1.1 Challenge

A cryptographic inventory should be viewed as an asset that supports an organization’s overall cybersecurity risk management strategy. A cryptographic inventory is needed to apply cryptographic policy across an organization’s digital infrastructures; to react quickly to security issues involving cryptography; and to inform transformations, such as migrating cryptography services to the cloud or deploying post-quantum cryptography.

This publication shares insights and findings about the use of cryptographic discovery tools to create a cryptographic inventory. This information may help organizations to start or improve their inventories. This publication is one of the first focused on practices for creating, expanding, or using a cryptographic inventory to support migration to quantum-resistant cryptographic technologies. As an initial draft, it offers: (1) a functional test plan that exercises the cryptographic discovery tools to determine baseline capabilities; (2) a use-case scenario to provide context and scope our demonstration; (3) the threats we will address in this demonstration; (4) a multifaceted approach for the discovery process that most organizations can start today; and (5) a high-level architecture based on our use case that integrates contributed discovery tools in our lab.

1.2 Outcomes

This preliminary practice guide can help your organization with the following:

- Identifying where and how public-key algorithms are being used in your information systems
- Raising internal awareness and understanding of risk-based cryptographic migration planning through the demonstration of tools, practice, and guidance
- Developing a risk-based playbook—involving people, processes, and technologies while connecting with existing risk management tools—for performing a migration to post-quantum cryptography

1.3 Preparing for the New Post-Quantum Standards

In recent years, there has been a substantial amount of research on quantum computers – machines that exploit quantum mechanical phenomena to solve mathematical problems that are difficult or intractable for conventional computers. If large-scale quantum computers are ever built, they will be able to break many of the public-key cryptosystems currently in use. This would seriously compromise the confidentiality and integrity of digital communications on the Internet and elsewhere.

In 2016, NIST began developing standards for post-quantum cryptography (also called quantum-resistant cryptography) to enable cryptographic systems that are secure against both quantum and classical computers and can interoperate with existing communications protocols and networks. New standards that specify key establishment and digital signature schemes that are designed to resist future attacks by quantum computers will be published in 2024.

Previous initiatives to update or replace installed cryptographic technologies have taken many years.

While the PQC standards are in development, organizations are encouraged to begin cryptographic discovery activities to identify the organization’s current reliance on quantum-vulnerable cryptography.
1.3.1 Public-Key Cryptographic Technologies

Cryptographic technologies are used throughout government, industry, and academia to authenticate the source and protect the confidentiality and integrity of information that we communicate and store. Cryptographic technologies include a broad range of protocols, schemes, and infrastructures, but they rely on a relatively small collection of cryptographic algorithms. They are mathematical functions that transform data, generally using a variable called a key to protect information. The protection of these key variables is essential to the continued security of the protected data.

Public-key (also known as asymmetric) cryptographic algorithms require the originator to use one key and the recipient to use a different but related key. One of these asymmetric keys, the private key, must be kept secret, but the other key, the public key, can be shared or even made public without degrading the security of the cryptographic process. Symmetric algorithms require a secret key to be shared by sender and receiver. The asymmetric (public-key) algorithms are used for data integrity (e.g., digital signature) and protected exchange of shared keys used by symmetric algorithms. The symmetric algorithms are more efficient for protection of bulk information, but the secure exchange and establishment of shared keys generally requires protection by asymmetric cryptography.

From time to time, the discovery of a cryptographic weakness, constraints imposed by dependent technologies, or advances in the technologies that support cryptanalysis make it necessary to replace a cryptographic algorithm. Practical quantum computing is a technological advance that will make the standard public-key algorithms now in use inadequate for provision of confidentiality of information, including secret keys. Most algorithms on which we depend are used worldwide in components of many different communications, processing, and storage systems. Cryptography has become ubiquitous. It is embedded in systems and components as different as operating systems, communications products, and Internet of Things devices in environments as different as space vehicles, automobiles, enterprise data centers, household appliances, and embedded medical devices. Information system owners are often unaware of what components have cryptography embedded or how the cryptography is used in each case.

Also, almost all information systems lack cryptographic agility—that is, they are not designed to encourage support of rapid adaptations of new cryptographic primitives and algorithms without making significant changes to the system’s infrastructure. As a result, an organization may not be able to easily alter or replace its cryptographic mechanisms when needed. While some components of some systems tend to be replaced by improved components on a relatively frequent basis (e.g., cell phones), other components are expected to remain in place for a decade or more (e.g., components in electricity generation and distribution systems).

Communications interoperability and records archiving requirements introduce additional constraints on system components. As a general rule, cryptographic algorithms cannot be replaced until all components of a system are prepared to process the replacement. Updates to protocols, schemes, and infrastructures often must be implemented when introducing new cryptographic algorithms. Consequently, algorithm replacement can be extremely disruptive and often takes a long time to complete. Unfortunately, the implementation of post-quantum public-key standards is likely to be even more problematic than the past introduction of new classical cryptographic algorithms. In the absence
of significant implementation planning, it is likely to be decades before the community replaces most of
the quantum-vulnerable public-key systems currently in use.

It would be ideal to have “drop-in” replacements for quantum-vulnerable algorithms (e.g., RSA and
Diffie-Hellman) for each of these purposes. Unfortunately, each of the new quantum-resistant
algorithms has at least one requirement for secure implementation that makes drop-in replacement
unsuitable sometimes. For example, the selected algorithms have large signature sizes, involve excessive
processing, require large public and/or private keys, require operations that are asymmetric between
sending and receiving parties and require the responder to generate a message based on the initiator’s
public value, and/or involve other uncertainties with respect to computational results. Depending on the
algorithm and the operation using that algorithm, secure implementation may need to address issues
such as public-key validation, public-key reuse, decryption failure even when all parameters are
correctly implemented, and the need to select new auxiliary functions (e.g., hash functions used in
digital signature schemes). Even where secure operation is possible, performance and scalability issues
may demand significant modifications to protocols and infrastructures.

Different post-quantum algorithms can have significantly different performance characteristics and
implementation constraints (with respect to key sizes, signature sizes, resource requirements, etc.).
Consequently, different algorithms can be more suitable than others for specific applications. For
example, the signature or key size might not be a problem for some applications but can be
unacceptable for others. Some widely used protocols need to be modified to handle larger signatures or
key sizes (e.g., using message segmentation). Implementations of new applications will need to
accommodate the demands of post-quantum cryptography (PQC) and the schemes developed that
incorporate PQC for digital signatures and key establishment. In fact, PQC requirements may actually
shape some future application standards. The replacement of algorithms generally requires changing or
replacing cryptographic libraries, implementation validation tools, hardware that implements or
accelerates algorithm performance, dependent operating system and application code, communications
devices and protocols, and user and administrative procedures. Security standards, procedures, and best
practice documentation are being changed or replaced, and the same will be needed for installation,
configuration, and administration documentation.

When a decision is made to replace an algorithm, it is necessary to develop a playbook that takes all of
the above factors into consideration. Some elements of the playbook are dependent on the
characteristics of both the algorithms being replaced and the replacement algorithms. Other elements
needed for developing a detailed migration playbook can be determined before the replacement
algorithms are selected and documented—for example, discovery and documentation of systems,
applications, protocols, and other infrastructure and usage elements that use or are dependent on the
algorithms being replaced.

The first step in PQC migration planning is to identify where and for what purpose public-key
cryptography is currently being used. Public-key cryptography has been integrated into existing
computer and communications hardware, operating systems, application programs, databases,
communications protocols, key infrastructures, and access control mechanisms. Examples of public-key
cryptography uses include:

- Digital signatures used to provide source authentication and integrity authentication as well as
  support the non-repudiation of messages, documents, or stored data
Identity authentication processes used to establish an authenticated communication session or authorization to perform a particular action

Key transport of symmetric keys (e.g., key-wrapping, data encryption, message authentication keys) and other keying material (e.g., initialization vectors)

This volume focuses on meeting the challenge of discovering where and how an enterprise’s public-key cryptography is used.

### 1.4 Demonstration Activity

To support the first step in migrating to post-quantum algorithms — identifying where and for what purpose public-key cryptography is being used within an enterprise — this project will document a demonstration activity which leverages discovery tools and platforms. Identifying assets such as hardware and software as part of an inventory is a core function of the Cybersecurity Framework (CSF) and a basic pre-condition for any organization to effectively manage cybersecurity risk. This project extends an existing inventory capability by identifying cryptographic assets and subsequently correlating them to hardware, software, and services that have been previously inventoried.

This project takes a holistic approach for the discovery of cryptographic assets by using a combination of active and passive discovery techniques that are described in detail in this document. In summary, we focus on vulnerable cryptographic algorithms within an organization’s digital systems and codebases, and use the discovered cryptographic assets to support the prioritization of their replacement. Lab demonstrations aim to leverage existing enterprise tools (code repositories, Governance, Risk, and Compliance [GRC] platforms, SIEM solutions, configuration management databases, etc.) and will use multiple commercially available and open-source discovery tools to achieve project objectives.

We have selected an interchange format to demonstrate using the output from the discovery tools for post-discovery risk analysis. We invite implementers to use the interchange format as part of an organization’s approach to leveraging discovery results.

### 2 How to Use This Guide

This NIST Cybersecurity Practice Guide focuses on one identified practice to ease migration from the current set of public-key cryptographic algorithms to replacement algorithms that are resistant to quantum computer-based attacks. It is an initial, preliminary public draft that shares what has been learned to-date regarding the use of automated cryptographic algorithm discovery tools.

Readers in IT roles such as security architects and system administrators who are responsible for monitoring the state of implementation of PQC standards in technology to inform their migration plan may also want to refer to a separate document in this series: NIST SP 1800-38C: Quantum-Resistant Cryptography Technology Interoperability and Performance Report. That report summarizes the outcomes from the Performance and Interoperability Workstream testing, in which we identified the challenging problems and bottlenecks that integrators will face when transitioning systems to post-quantum-ready algorithms.

This guide and NIST SP 1800-38C were preceded by NIST SP 1800-38A: Executive Summary, Migration to Post-Quantum Cryptography: Preparation for Considering the Implementation and Adoption of Quantum Cryptography.
Safe Cryptography. Business decision makers, including chief information security and technology officers should refer to this document to understand the drivers for this project and the challenges we plan to address.

This guide uses a monospace typeface for JavaScript Object Notation (JSON) examples.

3 Approach: The Migration to Post-Quantum Cryptography Project’s Contribution to Quantum Readiness

A successful post-quantum cryptography migration will take time to plan and conduct and can be described by a multistep approach. The approach is consistent with the activities below from the Quantum-Readiness: Migration to Post-Quantum Cryptography factsheet [4] created in partnership with the Department of Homeland Security’s Cybersecurity & Infrastructure Security Agency (CISA) and the National Security Agency (NSA).

1. Establish a Quantum-Readiness Roadmap
2. Prepare a Cryptographic Inventory
3. Discuss Quantum Safe Roadmaps with Technology Vendors
4. Determine Supply Chain Quantum-Readiness

The findings of the discovery activity and subsequent interoperability and performance activity can be used in refining organizations’ quantum-readiness roadmaps as the scope, distribution, characteristics, and requirements of their quantum-vulnerable implementations emerge.

One aspect that the NCCoE Migration to Post-Quantum Cryptography Project approach focuses on is automated tools that address step 2, Prepare a Cryptographic Inventory. This step should be taken to identify vulnerable cryptographic algorithms used by an organization. Automated tools should identify the cryptographic algorithms used in hardware and software modules, libraries, and embedded code. Automated tools should also identify the cryptographic algorithms currently used by an enterprise to support cryptographic key establishment and management underlying the security of cryptographically protected information and access management processes, as well as algorithms used to protect the source and content integrity of data at rest, in transit, and in use.

After the vulnerable public-key cryptography components and associated assets in the enterprise are identified, the next objective of the project is to prioritize those components that need to be considered first in the migration using a risk management methodology informed by the sensitivity and criticality of the information being protected over time.

In 2021 NIST invited organizations to provide letters of interest describing products and technical expertise to support the Migration to Post-Quantum Cryptography project. This notice was the initial step for the NCCoE in collaborating with technology companies to address cybersecurity challenges identified under this project. Commercial and open-source software and hardware technology providers responded with technology and tools that can provide organizations a head start to migrate to post-quantum cryptography and signed Cooperative Research and Development Agreement (CRADA) with
NIST to become project collaborators. The project collaborators first met in June 2022 and established two workstreams, each of which focuses on a specific aspect of migration to PQC.

This initial draft incorporates the contributions of consortium members in terms of their technologies and also offers initial strategies on how to leverage discovery to prioritize migration technologies.

Updates to this document will be made when additional demonstrations are completed. Section 6 discusses areas we have identified for future work in this workstream.

3.1 Audience

This document shares insights for medium to large enterprises using cryptographic technologies (products and services including cloud services) that include quantum-vulnerable public-key cryptographic algorithms and for companies that supply services and products that employ quantum-vulnerable public-key cryptographic algorithms. Within these audiences, we focus on individuals supporting those responsible for system provisioning, maintenance, and security. This audience may include business decision makers such as authorizing officials and data owners and operators who are concerned with cryptographic risk analysis capabilities.

Additionally, U.S. Federal agencies may find the capabilities and products that we demonstrate useful for assigned tasks in the November 2022 Office of Management and Budget (OMB) Memorandum on Migrating to Post-Quantum Cryptography (M-23-02) [5]. This memorandum includes steps for agencies to take to transition to PQC.

3.2 Scope

This publication includes an example scenario to frame the challenge of cryptographic discovery from the perspective of an organization beginning its migration to PQC. The scenario also scopes the desired outcomes. In Section 3.2.2, we discuss the approaches we will take to realize the outcomes described in Section 3.2.1. In Section 3.2.3, we specify the cryptographic algorithms that are considered vulnerable in the context of this demonstration.

3.2.1 Example Discovery Scenario for a Medium-Sized Business

The following scenario describes a fictional company and its approach to migration to PQC.

ZetaMSB Inc. (Zeta) is a medium-sized business that provides IT consultancy services regionally. Zeta has approximately 1000 employees with two office locations; however, a quarter of the employees are primarily remote workers. Further, each employee is issued an organizationally managed laptop to access proprietary Zeta data to support IT consultancy clients. This proprietary knowledge base is stored on a mixture of leased bare metal servers that Zeta manages and cloud services. Zeta also has a development team that creates software and services which support its clients and Zeta internal processes.

Zeta’s CISO has recently been tasked with coming up with a strategy to protect Zeta’s critical information and its competitive edge far into the future. To that end, the forward-looking CISO has started to read about the potential future impacts of quantum computing – specifically the threats to the integrity and confidentiality of data. The CISO consequently becomes concerned about Zeta’s current cryptographic posture and how competitors or other bad actors could use cryptanalytically...
relevant quantum computers (CRQCs) to exfiltrate proprietary data while stored on Zeta’s servers or in transit to their remote workers.

Zeta has a fairly mature cybersecurity program that leverages the NIST Cybersecurity Framework (CSF), and as such, periodically identifies future threats as part of risk management. As a result, the CISO decides to develop a plan that will address CRQC threats to their organization by incorporating into their long-term cyber strategy the migration of their systems and services that use vulnerable cryptography to quantum-resistant cryptography, hereafter referred to as post-quantum cryptography (PQC). The CISO adopts a phased approach covering multiple years due to their products having dependencies on third-party software that does not yet incorporate PQC. However, they conclude that the initial phase could be started immediately by expanding current hardware and software inventory processes to identify quantum-vulnerable cryptography in use within Zeta.

In consultation with stakeholders across the organization, the CISO derives a three-pronged approach for identifying the uses of vulnerable cryptography within the organization. First, Zeta’s newly formed migration team will identify digital cryptographic assets on operational systems fielded to Zeta employees and on systems and devices that are managed by Zeta’s IT department. These include operating systems, communications servers and controllers for wired and wireless internal and external networks, third-party application software, application software developed in-house, security systems such as firewalls, and key generation and management systems. In addition, employees may use cloud services that are not managed by Zeta’s IT department, but that provide cryptographic protection for Zeta information. The migration team concludes the best way to identify these services is to capture traffic from operational networks and examine it for instances of vulnerable cryptography usage with a focus on the information that traverse the public internet. Finally, to align with the organization’s “shift-left” methodology, the migration team recommends identifying vulnerable cryptography used by their DevOps team in their codebases. This would enable the DevOps team to update their codebases with post-quantum cryptography. After documenting this approach, the CISO directs the migration team to automate as much of the discovery process as possible, leveraging existing enterprise services if feasible.

The Zeta migration team proceeds to perform market research on vulnerable cryptography discovery tools available in commercial and open-source products. They conclude that multiple products may be necessary to achieve their desired outcomes. That presents a challenge to the team, however, because there is a discrete reporting format for each discovery platform to provide a common data format for collecting the data. As a result, to support automation, each discovery platform’s reporting format will need to be normalized into a common format in order to integrate into existing security event aggregation and reporting tools.

Resources for the migration task are limited at Zeta because of other planned IT modernization efforts occurring in the same timeframe. As a result, the CISO must follow a risk-based approach to prioritize which systems will be migrated first after the inventory has been completed. Fortunately, Zeta has identified processes and knowledge assets that are critical to the viability of its business by way of past CSF activities. These processes and assets will be mapped to operational systems and services identified by the discovery platforms and given higher priority in the migration effort.
3.2.2 Project Execution of Example Scenario in Our Lab

To create an implementation in our NCCoE Migration to PQC lab environment for the scenario above, the NCCoE team is collaborating with consortium members to create an appropriate environment including vulnerable algorithm discovery tools, representative enterprise infrastructure, and test data. The enterprise infrastructure includes virtual and hardware endpoint computing devices, such as laptops, with common software installed such as web browsers and business productivity software. This will be complemented by servers that leverage cryptographic services, such as HTTPS and Secure Shell (SSH). The DevOps environment will be built using common components that are described later in this document.

Finally, test data will include captured network traffic that has protocols using known vulnerable cryptography. These captures can support scenarios in which network traffic is “replayed” to simulate an operational network, where network-based sensors for discovering vulnerable cryptography can perform an analysis in real time. It also supports an alternative discovery scenario where an organization has historical captures that are uploaded to the discovery tool for analysis. The captures will function as repeatable “known answer tests” where the inputs (vulnerable cryptography) are known and should be detected by the discovery tools.

Additional test data will include a representative code base that leverages classical cryptography that is scanned for vulnerable cryptography usage in two scenarios:

- Code that exists in a cloud and on-premises code repository
- A project in active development where code is scanned before each pull or merge request

The first scenario allows for analysis of existing codebases, while the second detects and flags potential vulnerabilities before developer code is merged into the main code branch. As a result, the flagged code is reviewed and the individual who is responsible for maintaining the repository directs the developer to modify the pull request such that quantum-resistant code (or libraries) are implemented. Once the changes are made to the satisfaction of the maintainer, the pull request is approved, and the DevOps process proceeds.

With the preceding context, we’ve decomposed the discovery of vulnerable algorithms into the following use cases:

- Vulnerable cryptography used in code, compiled binaries, or dependencies during a continuous integration/continuous delivery (CI/CD) development pipeline
- Vulnerable cryptography used in assets on end-user systems and servers, to include applications and associated libraries
- Vulnerable cryptography used in network protocols, enabling traceability to specific systems using active scanning and historical traffic captures

The remainder of this section discusses each use case, including the scope and existing best practices that will be leveraged.
3.2.2.1 Protecting the Code Development Pipeline

Protecting the code development pipeline in the context of migration to PQC is closely related to the outcomes listed for the NCCoE DevSecOps project [6], which describes DevSecOps as helping to “ensure that security is addressed as part of all DevOps practices by integrating security practices and automatically generating security and compliance artifacts throughout the processes and environments, including software development, builds, packaging, distribution, and deployment.” The DevSecOps project leverages the Secure Software Development Framework (SSDF) [7] to define the tasks that can be implemented as part of a DevSecOps approach. Similarly, we align our migration tasks with the framework provided by the SSDF.

The SSDF defines secure software development practices that are organized into four practice groups shown in Table 1. This demonstration will focus on the Protect the Software (PS), Produce Well-Secured Software (PW), and Respond to Vulnerabilities (RV) groups.

Table 1 SSDF Practice Groups

<table>
<thead>
<tr>
<th>Practice Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the Organization (PO)</td>
<td>Organizations should ensure that their people, processes, and technology are prepared to perform secure software development at the organization level. Many organizations will find some PO practices to also be applicable to subsets of their software development, like individual development groups or projects.</td>
</tr>
<tr>
<td>Protect the Software (PS)</td>
<td>Organizations should protect all components of their software from tampering and unauthorized access.</td>
</tr>
<tr>
<td>Produce Well-Secured Software (PW)</td>
<td>Organizations should produce well-secured software with minimal security vulnerabilities in its releases.</td>
</tr>
<tr>
<td>Respond to Vulnerabilities (RV)</td>
<td>Organizations should identify residual vulnerabilities in their software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future.</td>
</tr>
</tbody>
</table>

Table 2 maps our project security characteristics to tasks in the SSDF.

Table 2 Security Characteristic Mapping to SSDF Tasks

<table>
<thead>
<tr>
<th>SSDF Task ID</th>
<th>Task Description</th>
<th>Migration to PQC Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS.3.2</td>
<td>Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]).</td>
<td>Creation of a cryptography bill of materials (CBOM)</td>
</tr>
<tr>
<td>PW.6.1</td>
<td>Use compiler, interpreter, and build tools that offer features to improve executable security.</td>
<td>Inspection of source code, dependent libraries, and containerized software via continuous integration actions</td>
</tr>
</tbody>
</table>
As noted in Table 2, this demonstration will experiment with the creation of a cryptographic bill of materials (CBOM). CBOMs are related to and build upon a software bill of materials (SBOM), which is a digital record containing the details and supply chain relationships of various components used in building software. A CBOM extends an SBOM by defining an object model to describe cryptographic assets and their dependencies. CBOMs, when integrated into reporting and software development practices, have the potential to enable organizations to manage and report usage of cryptography, benefiting asset inventory activities. The CBOM model is still developing and may not address every use case for all organizations, and in future drafts of this publication we will describe our approach to automating the creation of CBOMs or other formats as well as incorporating CBOMs into a broader asset inventory effort.

### 3.2.2.2 Operational Systems and Applications

The list below is a non-exhaustive set of typical cryptographic assets that are in scope for this demonstration based on the capabilities exercised in the lab environment thus far. Note that individual discovery platforms may have extended capabilities. The list is sectioned into two categories: executable and non-executable file objects. Executables are components of runnable software, and non-executables are filesystem objects referenced by executables. Non-executables include keystores and other formats that may contain both public and private asymmetric keys. This example implementation detects and reports public/private keypairs which implement one of the vulnerable algorithms listed in Table 3. Note that formats may support password-based encryption of private keys, which may affect the fidelity of the discovery results.

- **Executables**
  - Application binaries
  - Cryptographic libraries
  - Java archives

- **Non-executables**
  - Keystores: [PKCS#12](#), [Java Keystores](#), [Key Data Sets](#)
  - Other key formats: [OpenPGP Keys](#), [X.509 Certificates](#), [OpenSSH Keys](#), [PKCS#1](#), [PKCS#8](#)
3.2.2.3 Transport Protocols and Network Services

The Canadian National Quantum Readiness Best Practices and Guidelines document [8] suggests a number of technology protocols that organizations should scan for vulnerable cryptography. For this demonstration, based on exercising the discovery platforms in our lab environment, our initial scope is three core protocols—Transport Layer Security (TLS), Secure Shell (SSH), and Internet Protocol Security (IPsec). We chose these protocols due to their ubiquitous enterprise usage. TLS is a secure tunneling protocol widely implemented in browsers and web servers. TLS is used to protect multiple application layer protocols, including HTTPS, SMTP (via STARTTLS), IMAPS, POP3S, and Microsoft Remote Desktop Protocol (RDP). Contemporary versions of TLS make use of X.509 certificates and corresponding private keys which rely on quantum-vulnerable signature and key exchange algorithms. The SSH protocol enables a user or an automated process to remotely access the shell of a server system. Like TLS, asymmetric cryptographic keys are used to authenticate and establish encrypted SSH connections.

Finally, IPsec enables cryptographic-based security for IPv4 and IPv6 and is typically used in virtual private networks (VPNs).

3.2.3 Vulnerable Cryptographic Algorithms

We have based our vulnerable algorithm discovery on the NIST Post-Quantum Cryptography standardization process. Figure 1 represents the NIST cryptographic standards and guidelines, which includes public key (asymmetric) signature and key establishment schemes. Symmetric cryptographic primitives, such as block ciphers and hash functions, are not as drastically impacted by the advent of a (cryptographically relevant) quantum computer [9] and are not in scope for this project. This scoping provides the basis to develop a discovery policy across disparate platforms.
In this demonstration, the algorithms listed in Table 3 are considered vulnerable.

**Table 3 Scope of Vulnerable Algorithms**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic Curve Diffie Hellman (ECDH) Key Exchange</td>
<td>Asymmetric algorithm for digital signatures/key exchange</td>
<td>NIST SP 800-56A/B/C</td>
</tr>
<tr>
<td>Menezes Qu Vanstone (MQV) Key Exchange</td>
<td>Asymmetric algorithm for key exchange</td>
<td>NIST SP 800-56A/B/C</td>
</tr>
<tr>
<td>Elliptic Curve Digital Signature Algorithm (ECDSA)</td>
<td>Asymmetric algorithms for digital signatures/key exchange</td>
<td>FIPS PUB 186-5</td>
</tr>
<tr>
<td>Diffie Hellman (DH) Key Exchange</td>
<td>Asymmetric algorithms for digital signatures</td>
<td>IETF RFC 3526</td>
</tr>
<tr>
<td>RSA Encryption Algorithm</td>
<td>Asymmetric algorithms for digital signatures/key establishment</td>
<td>SP 800-56B Rev. 2</td>
</tr>
<tr>
<td>RSA Signature Algorithm</td>
<td>Asymmetric algorithms for digital signatures/key exchange</td>
<td>FIPS PUB 186-5</td>
</tr>
</tbody>
</table>
## 3.3 Terminology

This document and future volumes will align with the terminology defined in IETF’s Terminology for Post-Quantum Traditional Hybrid Schemes [10]. The document is intended to be used as a reference and to ensure consistency and clarity across different protocols, standards, and organizations. In particular, the remainder of this document references the following terms, followed by their definitions.

**Traditional or Classical Cryptographic Algorithm:** An asymmetric cryptographic algorithm based on integer factorization, finite field discrete logarithms, or elliptic curve discrete logarithms.

**Post-Quantum Cryptographic Algorithm:** An asymmetric cryptographic algorithm that is believed to be secure against attacks using quantum computers as well as classical computers. The algorithms identified as post-quantum algorithms are defined in:

- **FIPS 203** (Draft), Module-Lattice-Based Key-Encapsulation Mechanism Standard [11], specifies a cryptographic scheme called “Module Learning with errors Key Encapsulation Mechanism, or MLWE-KEM,” which is derived from the CRYSTALS-KYBER algorithm.

- **FIPS 204** (Draft), Module-Lattice-Based Digital Signature Standard [12], specifies the “Module Learning with Errors Digital Signature Algorithm, or ML-DSA,” which is based on the CRYSTALS-Dilithium submission.

- **FIPS 205** (Draft), Stateless Hash-based Digital Signature Standard [13], specifies the “Stateless Hash-based Digital Signature Algorithm, or SLH-DSA,” which is based on the SPHINCS+ submission.

These initial quantum-resistant algorithms and stateless hash-based signature standard will augment the public-key cryptographic algorithms already contained in FIPS 186-5, Digital Signature Standard (DSS) [14], as well as SP 800-56A Revision 3, Recommendation for Pair-Wise Key- Establishment Schemes Using Discrete Logarithm Cryptography [15], and SP 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography [17]. Additional quantum-resistant algorithms will be defined as the standardization effort progresses.

Finally, several members of our consortium have moved to use the terms “quantum-safe” or “quantum-ready” as opposed to “post-quantum” to refer to the algorithms and technologies that are protected against cryptographically relevant quantum computer (CRQC) attacks. This document continues to use “post-quantum” and will be updated as these terms evolve.

## 3.4 Risk Assessment

In this section we discuss threats to classical algorithms and the specific attacks that researchers have determined are feasible when a CRQC is operational. Next, we consider the vulnerabilities that a CRQC...
will introduce with a discussion of a proposed Common Weakness Enumeration (CWE) that organizations could potentially use during their internal prioritization efforts. Finally, we discuss existing research efforts to develop a risk methodology that have influenced this project’s approach to demonstrating a migration prioritization list.

3.4.1 Threats to Classical Cryptography

At this time, there are two quantum algorithms that are projected to be used in attacks against data protection schemes such as digital encryption and signatures. The first is Shor’s algorithm, which provides an efficient method for computing the discrete-logarithm problem and the elliptic curve discrete-logarithm problem, as well as the problem of factoring large integers, thus breaking current key exchange, digital signature, and public key encryption methods that are based on asymmetric cryptography. Because of Shor’s algorithm, new cryptographic algorithms are needed that are resistant to attacks that can be launched from both classical and quantum computers. The second quantum algorithm is Grover’s algorithm, which can be used to speed up the identification of a secret key in a key address space. To thwart this type of attack, strong encryption algorithms are needed with keys whose key address space is large enough to be considered not vulnerable.

Digital encryption and signature schemes are widely used in transport protocols like (D)TLS, IKEv2/IPsec, QUIC, and SSH. They all include a key exchange phase where the peers exchange asymmetric keys which enable them to establish a shared secret using ECDH. They then proceed to derive a symmetric key which is used to symmetrically encrypt data exchanged between the peers. The Authentication phase includes providing an asymmetric signature of a transcript of the exchanged data which proves that the peer signed this data with its private key. The corresponding public key is usually included with the identity of the peer and is authenticated by using PKI or other methods. That way the peers can verify they are talking to the peer with the expected identity who holds the expected public key.

A quantum computer could break the asymmetric schemes for key exchange and signing. Shor’s algorithm could break ECDH, which means that a quantum-capable threat actor could recover the symmetric key used to encrypt data. Specifically, for encryption, although there is no CRQC today, someone could be storing data encrypted with TLS or other protocols today in order to retroactively decrypt them in the future with a quantum computer, more commonly known as harvest and decrypt or store now decrypt later attacks.

Consequently, protection of data is needed at rest, in transit, and in use. As a reference we have listed below a variety of techniques an adversary can use to enable the harvesting of encrypted data today, as documented in MITRE’s ATT&CK framework [16], a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. Further, it may not be necessary for the adversary to own their own CRQC, as it is expected that adversaries will have cloud access to CRQCs in the future to conduct their attacks.

- **Adversary-in-the-Middle** - Adversaries may attempt to position themselves between two or more networked devices using an adversary-in-the-middle (AitM) technique to support follow-on behaviors such as network sniffing or transmitted data manipulation. By abusing features of common networking protocols that can determine the flow of network traffic (e.g., ARP, DNS, LLMNR), adversaries may force a device to communicate through an adversary-controlled system so they can collect information or perform additional actions.
Automated Collection - Once established within a system or network, an adversary may use automated techniques for collecting internal data. Methods for performing this technique could include use of a command and scripting interpreter to search for and copy information fitting set criteria such as file type, location, or name at specific time intervals. In cloud-based environments, adversaries may also use cloud APIs, command line interfaces, or extract, transform, and load (ETL) services to automatically collect data. This functionality could also be built into remote access tools.

Data from Cloud Storage - Adversaries may access data from improperly secured cloud storage.

Data from Local System - Adversaries may search local system sources, such as file systems and configuration files or local databases, to find files of interest and sensitive data prior to exfiltration.

Data from Network Shared Drive - Adversaries may search network shares on computers they have compromised to find files of interest. Sensitive data can be collected from remote systems via shared network drives (host shared directory, network file server, etc.) that are accessible from the current system prior to exfiltration. Interactive command shells may be in use, and common functionality within cmd may be used to gather information.

For a more in-depth discussion on threats to existing security architectures which employ vulnerable cryptography, we recommend readers review the content produced by other organizations such as the Accredited Standards Committee’s Quantum Computing Risks to the Financial Services Industry [18] and the Quantum-Readiness Working Group’s (QRWG) of the Canadian Forum for Digital Infrastructure Resilience (CFDIR) Best Practices and Guidelines [8].

3.4.2 Vulnerabilities

During some workstream discussions, the project team identified the need to report quantum vulnerabilities to existing upstream risk management systems in a standardized format. We chose Common Weakness Enumeration (CWE) [19], a community-developed list of software and hardware weakness types, due to its widespread use in industry. It serves as a common language and a baseline for weakness identification, mitigation, and prevention efforts.

CWEs in the cryptographic domain are currently categorized as a failure in a protection mechanism (CWE-693). These include existing CWEs (CWE-327: Use of a Broken or Risky Cryptographic Algorithm and CWE-326: Inadequate Encryption Strength, for example) that address weak algorithms.

The CWE concept is targeted around existing vulnerabilities. Future vulnerabilities such as those coming from a quantum threat do not appear to easily fit into this concept. Weaknesses only manifest if certain conditions are met, leading to the potential of many false positives. Nevertheless, a major benefit in using the CWE approach would be to signal future issues in a way that allows remediation development to be planned for in advance as a feature, rather than later as a remediation action. This could be handled in different ways within the CVE scheme:

- through the use of additional attributes,
- through defining a separate block of codes for future weaknesses, or
- through logic in evaluating additions to existing blocks.
During project workstream meetings, a proposed CWE was created (see Table 4) to capture the particular weakness of algorithms that are quantum-vulnerable but are otherwise safe. Due to the nature of this weakness, not all proposal elements were applicable. It was proposed to be a Child of CWE-327 (CWE-327: Use of a Broken or Risky Cryptographic Algorithm) with this description:

- Cryptographic algorithms are used to protect the confidentiality and authenticity stored in or transmitted through an untrusted medium. Quantum-vulnerable algorithms will be exploitable if or when a cryptographically relevant quantum computer (CRQC) is built, in which case sensitive information could be exposed, data could be undetectably modified, and the identities of users and devices could be spoofed, among other impacts.

- Data that is encrypted by a quantum-vulnerable encryption or key establishment algorithm and has a long data protection period is especially at risk, because an adversary can store encrypted data today and decrypt it later with a CRQC. Such an attack is called store-now-decrypt-later, and it creates significant risk for long-term data confidentiality. Additionally, an adversary with a CRQC could forge signatures for quantum-vulnerable algorithms, creating a risk for signatures with a long data protection period, especially those used to sign software images in devices where the root of trust cannot be upgraded.

Table 4 Submitted CWE

<table>
<thead>
<tr>
<th>Elements to Include</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Use of a quantum-vulnerable algorithm</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>The use of a quantum-vulnerable algorithm is a risk that may result in the exposure of sensitive information if and when a cryptographically relevant quantum computer becomes available.</td>
</tr>
<tr>
<td><strong>Extended Description</strong></td>
<td>The use of a quantum-vulnerable algorithm creates a risk if a cryptographically relevant quantum computer (CRQC) becomes available. A CRQC could threaten all public key algorithms based in the integer factorization and (elliptic curve) discrete logarithm problem and compromise whatever data has been protected. In particular, an adversary storing encrypted data today could theoretically decrypt them later with a CRQC. Such an attack is called harvest-now-decrypt-later, and it creates significant risk for long-term data confidentiality. Additionally, an adversary with a CRQC could forge signatures for quantum-vulnerable algorithms, creating risk for signatures with a long data protection period.</td>
</tr>
<tr>
<td><strong>Modes of Introduction</strong></td>
<td>Architecture and Design</td>
</tr>
<tr>
<td><strong>Potential Mitigations</strong></td>
<td>Use standardized quantum-safe asymmetric algorithms or hybrid schemes that incorporate both post-quantum and traditional asymmetric algorithms.</td>
</tr>
<tr>
<td><strong>Common Consequences</strong></td>
<td>Confidentiality</td>
</tr>
<tr>
<td></td>
<td>Technical Impact: Read Application Data</td>
</tr>
<tr>
<td></td>
<td>The confidentiality of sensitive data may be compromised by the use of a quantum-vulnerable cryptographic algorithm.</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
</tbody>
</table>
Elements to Include | Value
---|---
**Technical Impact: Modify Application Data**<br>The integrity of sensitive data may be compromised by the use of a quantum-vulnerable cryptographic algorithm.

**Accountability**
Non-Repudiation
**Technical Impact: Hide Activities**
If the cryptographic algorithm is used to ensure the identity of the source of the data (such as digital signatures), then the use of a quantum-vulnerable algorithm will compromise this scheme and the source of the data cannot be proven.

<table>
<thead>
<tr>
<th>Applicable Platforms</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstrative Examples</td>
<td>N/A</td>
</tr>
<tr>
<td>Observed Examples</td>
<td>N/A</td>
</tr>
<tr>
<td>Relationships</td>
<td>N/A</td>
</tr>
<tr>
<td>References</td>
<td>The references provided were to NSA Commercial National Security Algorithm Suite 2.0 (<a href="https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF">https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF</a>) and NIST Post-Quantum Cryptography (<a href="https://csrc.nist.gov/projects/post-quantum-cryptography">https://csrc.nist.gov/projects/post-quantum-cryptography</a>).</td>
</tr>
</tbody>
</table>

At the time of writing, the proposal has been submitted to the CWE team for feedback.

### 3.4.3 Survey of Risk Methodologies

This project aims to leverage a reusable, generalized approach that could facilitate the migration process for many organizations. The following subsections describe several risk methodologies specifically to address the threat of a CRQC at the time of writing. The methodologies are similar in that they all describe a cryptographic asset discovery activity from which a prioritization list can be derived. However, some of the methodologies describe sector-specific guidance to perform a risk assessment.

#### 3.4.3.1 Mosca’s Theorem

Mosca’s Theorem is commonly referenced as the starting point for organizations initiating a migration to post-quantum algorithms. Once the public-key cryptography components and associated assets in the enterprise are identified, the next element of the scope of the project is to prioritize those components that need to be considered first in the migration using a risk management methodology informed by Mosca’s Theorem and other recommended practices. The Global Risk Institute describes [20] one such approach below:
Phase 1 - Identify and document information assets, and their current cryptographic protection.

Phase 2 - Research the state of emerging quantum computers and quantum-safe cryptography. Estimate the timelines for availability of these technologies. Influence the development and validation of quantum-safe cryptography.

Phase 3 - Identify threat actors and estimate their time to access quantum technology “z”.

Phase 4 - Identify the lifetime of your asset’s “x”, and the time required to transform the organization’s technical infrastructure to a quantum-safe state “y”.

Phase 5 - Determine quantum risk by calculating whether business assets will become vulnerable before the organization can move to protect them. \((x + y > z)\)

Mosca’s Theorem has been applied in the report Preparing for Post-Quantum Critical Infrastructure by the Rand Corporation [21]. The report performed high-level assessments of quantum vulnerabilities in the 55 national critical functions (NCFs). The report found six NCFs are high priority for assistance in migration, 15 are medium priority, and 34 are low priority.

3.4.3.2 Crypto Agility Risk Assessment Framework (CARAF)

The Crypto Agility Risk Assessment Framework (CARAF) [22] provides an extension to Mosca’s Theorem by focusing on developing information systems that encourage support of rapid adaptations of new cryptographic primitives and algorithms without making significant changes to the system’s infrastructure, otherwise known as cryptographic agility [1]. There are five phases to the framework summarized below:

- Phase 1 – identify threats
- Phase 2 – inventory of assets
- Phase 3 – risk estimation
- Phase 4 – secure assets through risk mitigation
- Phase 5 – organizational roadmap

Phases 1-3 are similar to Mosca’s algorithm in that the output of the risk analysis is a probability based on a timeline, given a threat such as the development of a quantum-capable system. Phases 4 and 5 aim to “facilitate informed decision making to accept, mitigate, or reject the risk from lack of crypto agility as well as plan to address the risk when appropriate.”

3.4.3.3 Financial Sector (FS) – ISAC Risk Model

The FS-ISAC Post-Quantum Cryptography Working Group has developed an Infrastructure Inventory Technical Paper [23] which suggests that financial organizations develop a Cryptographic Agility Index (CAI). The CAI is described as a holistic view that reflects several specific points around prioritization, controls, business capabilities, vendors, mitigation, and implementation plans.

A notable differentiator in the CAI methodology is whether a cryptographic asset is organizationally developed or from a third-party vendor. In other words, an organization should incorporate the risk of the continued use of third-party software in the overall risk calculation. The CAI also suggests sector-specific services, such as the SWIFT banking system, are carefully considered in the migration risk calculation process.
### 3.4.3.4 Telecom Sector

In September 2023, the GSM Association published a white paper titled Guidelines for Quantum Risk Management for Telco [24] to put forward a methodology that supports telecommunication service providers and the extended telecommunication supply chain by using telecom-relevant use cases as an example. In this white paper, the authors analyze two existing quantum risk assessment methodologies – Mosca’s “x,y,z” quantum risk model described in Section 3.4.3.1 and CARAF described in Section 3.4.3.2. Drawbacks to each approach specific to the telecom sector are identified, and recommendations are provided to modify the frameworks to better align with telecom technical and regulatory constraints.

### 3.4.3.5 Cybersecurity and Infrastructure Security Agency (U.S. DHS/CISA)

The DHS roadmap for the transition to post-quantum algorithms [25] advises organizations to consider the following factors when evaluating systems during the risk assessment process:

1. Is the system a high-value asset based on organizational requirements?
2. What is the system protecting (e.g., key stores, passwords, root keys, signing keys, personally identifiable information, sensitive personally identifiable information)?
3. What other systems does the system communicate with?
4. To what extent does the system share information with federal entities?
5. To what extent does the system share information with other entities outside of your organization?
6. Does the system support a critical infrastructure sector?
7. How long does the data need to be protected?

These factors are geared towards U.S. Federal Government system owners; however, they can also be applied to the private sector, especially organizations that support operational critical infrastructure.

### 3.4.3.6 Cyber Physical Systems (CPS)

A report titled Quantum Computing Threat Modelling on a Generic CPS Setup [26] proposes a risk methodology for CPSs based on a Process of Attack Simulation and Threat Analysis (PASTA) threat-modeling exercise complemented with attack trees and the STRIDE model for identifying threats. The report concluded that a threat-based model is a valid approach when assessing risk in the context of CPSs.
4 Architecture

The proposed project architecture is designed to align with the conceptual workflow depicted in Figure 2, based on lab discovery technology research and the use case described in Section 3.2.1. In Section 4.1, we present a systems-level view of an architecture that implements vulnerable algorithm discovery in the three areas within the scope of the project – the code development pipeline, operational network services and protocols, and operational systems and applications. Finally, we describe our approach to normalize the output from the discovery platforms into the common format. In future versions of this document, we will detail the architecture components that will ingest the normalized discovery platform output and produce a prioritization list.

4.1 Architecture Description

This section describes the “to be” architecture for each vulnerable algorithm discovery use case. This is subject to change as we continue to experiment with the platforms and receive feedback from the larger community of interest. This section may also help inform decision makers in your organization as use cases are developed to transition systems to PQC. In future revisions of this document, we will map these capabilities to existing cybersecurity best practices and describe a final architecture that will identify the usage of vulnerable algorithms within an organization and prioritize their replacement.

4.1.1 Protecting the Code Development Pipeline

Typical CI/CD pipeline components are described in NIST’s Implementation of DevSecOps for a Microservices-based Application with Service Mesh [27] and are reproduced in Table 5. In this example implementation, we focus on pipeline software, SDLC software, and repository components. This document describes the CI/CD processes at a high level, and the reader is encouraged to review the code development resources below for an in-depth review.
Table 5 Software Development Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline Software (CI)</td>
<td>Pulls code from a code repository, invokes the build software, invokes test tools, and stores tested artifacts to image registry</td>
</tr>
<tr>
<td>Pipeline Software (CD)</td>
<td>Pulls out artifacts, packages, and deploys the package based on computing, network, and storage resource descriptions</td>
</tr>
<tr>
<td>SDLC Software</td>
<td>Build tools (e.g., IDEs)</td>
</tr>
<tr>
<td></td>
<td>Testing tools (e.g., SAST, DAST, SCA)</td>
</tr>
<tr>
<td>Repositories</td>
<td>Source code repositories (e.g., GitHub)</td>
</tr>
<tr>
<td></td>
<td>Container image repositories or registries</td>
</tr>
<tr>
<td>Observability or Monitoring Tools</td>
<td>Logging and log aggregation tools</td>
</tr>
<tr>
<td></td>
<td>Tools that generate metrics</td>
</tr>
<tr>
<td></td>
<td>Tracing tools (sequence of application calls)</td>
</tr>
<tr>
<td></td>
<td>Visualization tools (combine data from above to generate dashboard/alerts)</td>
</tr>
</tbody>
</table>

Using the tools described in Table 5, we created a code development pipeline that integrates vulnerable algorithm discovery at the build stage. In the first scenario, an analyst can extract data from an existing codebase to create a queryable database. Once the database is created, the analyst creates and executes a query that detects the usage of vulnerable algorithms in the existing codebase. The query is executed from a command-line tool or an IDE plugin. The output is generated in SARIF format, described in Section 4.1.4.2, where it is consumed by the enterprise visualization platform.

We also demonstrate two automated versions of this process by integrating the query into an existing enterprise CI platform. In the first, an on-premises CI system analyzes the codebase using the command-line version of the tool as part of the build process. In the second, we use the built-in automation tools from a cloud-based code repository that are executed on each pull request [28]. The full code pipeline with vulnerable cryptography discovery activities described above are highlighted below as static application security testing in Figure 3.

Figure 3 Conceptual CI Pipeline
4.1.2 Operational Systems and Applications

Filesystem scanning sensors are used to detect quantum-vulnerable algorithms in software. An organization may choose to deploy scanners to end-user devices or servers through existing automated enterprise deployment tools. Scans are triggered either manually or through an automated mechanism, and the resulting output is transmitted to the back-end analysis engine or, in the manual case, uploaded to the back-end by the operator. This architecture demonstrates scanning of x64 Linux and Windows hosts, but other platforms may be supported by individual discovery platforms. Further, some discovery scanning solutions may offer integrations with endpoint detection and response (EDR) platforms, which combine real-time continuous monitoring and collection of endpoint data with rules-based automated response and analysis capabilities. Such integrations offer the benefit of leveraging already-existing cybersecurity processes and dashboarding capabilities. The scan is also performed on the binaries to discover algorithms that there might not be a source code for, as, for example, in third-party applications.

Figure 4 describes an architecture where the discovery platform sensor output is transmitted to the analysis engine.

Figure 4 Sensor data flow
4.1.3 Transport Protocols and Network Services

In the scenario described above, we identified two network discovery capabilities – real-time scanning and discovery, and “passive” scanning and discovery from historical packet captures. In Figure 5, we consider traffic capture from the cloud, on-premises, and untrusted networks. The cloud segment contains systems, both bare metal and virtual, that host internal corporate services. Network packets are mirrored to the cloud-based network traffic capture appliance; however, the implementation is dependent on the cloud provider. The network traffic is routed to the PQC vulnerable discovery platform via a secure tunnel to the on-premises network. The on-premises segment contains systems that also host internal services and additionally supports endpoint devices (e.g., laptops) that are operated by end users and managed by the organization. Here, network traffic is mirrored from a physical switch and routed to the network traffic capture appliance. Finally, network traffic from corporately managed endpoint systems that are operated by remote users cannot be directly forwarded to the PQC vulnerable discovery platform in real-time. In this scenario, network traffic is captured as a file and forwarded asynchronously to the discovery platform via a secure tunnel (e.g., VPN). Similarly, file-based historical network traffic captures are uploaded directly to the discovery platform for asynchronous analysis.

Figure 5 Passive Network Discovery
4.1.4 Common Output Elements for Identifying Vulnerable Systems

While discovery platforms can provide their own reporting capabilities that enable an administrator to quickly view the results of the discovery process, this demonstration integrates the reports from these platforms into the context of an enterprise-wide dashboard capability. Such an approach consolidates reports from disparate security products that an enterprise may already have fielded, such as EDR platforms. Further, consolidation of vulnerable algorithm discovery reports with other cybersecurity product reports enables enterprise-level risk management, rather than performing an analysis in isolation.

The reports produced by the discovery platforms in this demonstration are unique in that they do not use a common format for representing the discovery results. In a contrived example, a network discovery platform may identify a host system as `host.example.com:443`, whereas another may omit the port number (`host.example.com`). Therefore, we identified the need for a common format to represent normalized discovery reports, ideally by leveraging existing normalization efforts to the extent possible. The remainder of this section defines a preliminary version of a normalization scheme across all discovery platforms that would allow an organization to make a risk decision. It does not define a schema, but instead defines descriptive data elements and how they can be obtained from passive network observations, active network scans, endpoint monitoring, or configuration information, and how those elements can be compared.

In alignment with our project goal to support automation wherever feasible, the first step in our normalization effort was to identify structured machine-readable report formats, such as Extensible Markup Language (XML) or JavaScript Object Notation (JSON), that were supported by each discovery platform. From there, we identified the application programming interface (API) calls necessary to access the resulting report, if the discovery platform offered this capability. APIs are typically serviced via synchronous HTTP-based request/response paradigm.

4.1.4.1 Network Discovery Analysis

This section defines the data elements that are used to identify vulnerable algorithms used in protocols (as mentioned in Section 4.1.3) identified in Section 3.2.2. First, Table 6 below lists each data element and its corresponding representation format.

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP (v4 or v6) Address</td>
<td>String</td>
</tr>
<tr>
<td>Destination Port</td>
<td>Number</td>
</tr>
<tr>
<td>Hostname</td>
<td>String</td>
</tr>
<tr>
<td>Application Layer Protocol</td>
<td>String, Service Name or TLS ALPN ID</td>
</tr>
<tr>
<td>Application Software</td>
<td>String, Common Platform Enumeration (CPE) 2.3</td>
</tr>
<tr>
<td>Operating System</td>
<td>String, Common Platform Enumeration (CPE) 2.3</td>
</tr>
<tr>
<td>Device Vendor</td>
<td>String, Common Platform Enumeration (CPE) 2.3</td>
</tr>
</tbody>
</table>
Next, we describe how each field’s value is populated and note other constraints that may affect the translation of data presented in the discovery platform to our profile. Generally, *String* and *Number* data types are as defined in RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format* [29]. Service Names are as per RFC 6335 [30], Section 5.1, and are registered by the Internet Assigned Numbers Authority (IANA); the registry can be accessed at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. TLS Application Layer Protocol Negotiation (ALPN) Identifiers (ID) are defined in RFC 7301 [31] and registered at https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml#alpn-protocol-ids. CPE strings are as defined in the Common Platform Enumeration: Naming Specification, Version 2.3, NIST IR 7695 [32].

**IP Address Data Element Description**

In an IP packet containing a TLS Client Hello, the IP source address is that of the client. In an IP packet containing a TLS server response, the IP source address is that of the server. A packet contains a single IP address that corresponds to the client or the server.

In a PKIX certificate, a *subjectAltName ipAddress* field holds the address of the host associated with the certificate. Certificates most often are associated with servers. A certificate can contain multiple IP addresses.

A host observation of an address can contain multiple IP addresses due to multihoming.

**Destination Port Data Element Description**

In an IP packet containing a TLS Client Hello, the destination port is associated with the service. In an IP packet containing a TLS server response, the source port is associated with that service.

The destination port is essential in identifying a server. TLS is used to protect several protocols, and some protocols may appear on more than one port. For instance, the destination port 443 is registered for the HTTPS protocol, but HTTPS is also used by other registered protocols, including amt-soap-https, appserv-https, commtact-https, 26icros-https, llsurfup-https, oob-ws-https, oracleas-https, pcsync-https, pgpkey-https, plysrv-https, sun-sr-https, sun-user-https, tungsten-https, wap-push-https, wbem-exp-https, and wbem-https. Additionally, TLS is registered for use by several non-HTTPS protocols.

**Hostname Data Element Description**

A DNS hostname is a string of labels, each containing up to 63 octets, separated by dots (the ‘.’ Character), with a maximum total length of 255 octets [33]. A label is a string that starts and ends with an alphanumeric character and contains only alphanumeric characters and hyphens.

In a TLS Client Hello, the hostname of the server often appears in the Server Name extension, with the *NameType* of HostName. RFC 6066, Section 3 [34] requires this field to contain DNS hostnames and not string representations of addresses, though some clients send addresses in this field. The standard requires that no more than a single hostname appears in this extension.

In a TLS Server Certificate, the hostname of the server often appears in the *SubjectName* or *SubjectAltName*. Multiple hostnames can appear in the certificate.
The application layer protocol used in TLS is negotiated between the client and server. In a TLS Client Hello, the ALPN field contains a list of ALPN IDs. The server response contains a single ALPN ID. Some ALPN IDs are intentionally overloaded to hinder network monitoring; in particular, DNS over HTTPS (DoH) uses the ALPN ID of HTTPS. Thus, active scanning obtains more accurate measurements of the application layer protocols supported by a server, as compared to passive monitoring.

Application Software Data Element Description

Application software can be identified by a CPE string with a “part” of “a”. On a host, identification can be performed directly. On a network, it can be performed through banner scraping or behavioral fingerprinting.

Operating System Data Element Description

An operating system can be identified by a CPE string with a “part” of “o”. On a host, identification can be performed directly. On a network, it can be performed through banner scraping or behavioral fingerprinting.

Device Vendor Data Element Description

A hardware device can be identified by a CPE string with a “part” of “h”. On a host, identification can be performed directly. On a network, it can be performed through IEEE Organizational Unit Identifier (OUI) reporting or behavioral fingerprinting.

While the CBOM object model is still in its nascent stages, it could potentially provide the necessary structure and extensibility to capture the common data elements in a machine-readable format. Readers of this document are encouraged to reference the CycloneDX Bill of Materials specification, as it provided the basis for the CBOM structure. A CBOM defines a crypto-asset, which is a representation of a BOM component type. A crypto-asset is defined by its cryptoProperties object that describes predefined assetTypes. In the case of network discovery, we choose the protocol assetType which is described by the tlsCipherSuites property. The tlsCipherSuites property defines the TLS cipher suites supported by the TLS protocol instantiation reported by the discovery platform. Finally, we leverage the properties array, a name-value store defined in the core SBOM specification, to communicate the remainder of the common elements. An example follows below of a complete CBOM:

```json
{
    "bomFormat": "CycloneDX",
    "components": [
        {
            "bom-ref": "oid:1.3.18.0.2.32.104",
            "cryptoProperties": {
                "assetType": "protocol",
                "protocolProperties": {
                    "tlsCipherSuites": [
                        "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519)"
                    ]
                }
            }
        }
    ]
}
```
4.1.4.2 Static Analysis

In Sections 3.2.2.1 and 3.2.2.2, we identified a need to run vulnerable algorithm discovery tools to help scan and detect vulnerable code in source code, files, and databases. This demonstration normalizes the results in SARIF (Static Analysis Results Interchange Format) [35] as it defines a standard format for the
output of static analysis tools and can be used to streamline how static analysis tools share their results.

In this section, we specify our profile of the `run` object, which describes a single run of an analysis tool and contains the output of that run. Within the `run` object, a `tool` component describes the specific tool (in the context of this project, a discovery engine) which has completed a scan. Table 7 lists the properties of the `tool` component.

### Table 7 tool component property specification

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>mandatory</td>
<td>Consortium member-provided name</td>
</tr>
<tr>
<td>version</td>
<td>optional</td>
<td>SARIF includes many tool driver properties that can help to identify the tool version information that was used to create the SARIF log. It is recommended to use at least one of these properties to help identify the tool version, and therefore the rules version, at a later date.</td>
</tr>
<tr>
<td>downloadUri</td>
<td>optional</td>
<td>A URI from which this version of the tool can be downloaded.</td>
</tr>
<tr>
<td>informationUri</td>
<td>optional</td>
<td>A URI from which information about this version of the tool can be found.</td>
</tr>
<tr>
<td>Rules</td>
<td>optional</td>
<td>See “rules property” section below.</td>
</tr>
</tbody>
</table>

An example output of the SARIF `tool` property is presented below.

```json
    "tool": {
        "driver": {
            "name": "Discovery Platform Name",
            "organization": "Company Name",
            "product": "Product Name",
            "full_name": "Product Name 1.0.0.0",
            "version": "1.0.0.0",
            "semanticVersion": "1.0.0",
            "downloadUri": "https://download.uri.contoso.com/v.1.0",
            "informationUri": "https://information.uri.contoso.com/v.1.0",
            "rules": [...]
        }
    }
```

### Runs.tool.rules Property Description

The rules property encapsulates an array of zero or more `reportingDescriptor` objects, which contain the information describing a reporting item generated by the tool, which is either a) the result of the tool’s analysis or b) a notification encountered by the tool. Although there are many ways to define a SARIF log document, one recommended approach is to enumerate all the rules a tool has executed as a whole, even if there are no findings, then record all unique findings under the results property. This strategy will assist to identify what rules were executed but found nothing to report.
Table 8 runs.tool.rules property specification

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>mandatory</td>
<td>It is recommended that the id is a readable and stable identifier; it may be an opaque identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>optional</td>
<td>A localizable (optionally) string that represents an identifier that is understandable for the end user.</td>
</tr>
<tr>
<td>fullDescription</td>
<td>mandatory</td>
<td>A localizable object that describes in detail the reporting item. Use the text sub-property for a description string.</td>
</tr>
<tr>
<td>messageStrings</td>
<td>optional</td>
<td>A localizable object that defines the strings that will be generated when the tool execution happens. It is recommended to define the messageStrings/Default with a string so it can be used with the results property. For this project, we use strings with placeholders to unify as much as possible the elements of the quantum-vulnerable cryptography elements found. For details see the “Recommended Message Placeholder Elements” section.</td>
</tr>
<tr>
<td>helpUri</td>
<td>optional</td>
<td>An absolute URI of the primary documentation for the reporting item.</td>
</tr>
</tbody>
</table>

An example output of the SARIF run.tool.rules property is presented below.

```
"rules": [
{
  "id": "sqlqvc/mssql/qvs001",
  "name": "certificate detection",
  "fullDescription": {
    "text": "Full description."
  },
  "messageStrings": {
    "Default": {
      "text": "Vulnerable Algorithm: [{0}]. Certificate ID `[{1}]` with subject `[{2}]`, key length `{3}`, expiration date `[{4}]` that is using quantum-vulnerable cryptography was detected."
    }
  },
  "helpUri": "https://www.example.com/rules/sqlqvc/mssql/ qvs001"
}, ...
]
```

**Runs.results Property Description**

The results property is an array of result objects, which each represent a single result detected by the tool during a run. Following the idea of specifying all rules that were executed under the runs.tool.rules property, the properties in Table 9 are used to represent the actual findings in the tool execution.
Table 9 runs.results property specification

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleId</td>
<td>mandatory</td>
<td>Although the SARIF specification denotes that this property existence depends on many circumstances, we recommend using it. This property would match a value from the id property from an object defined under runs.tool.rules.</td>
</tr>
<tr>
<td>message</td>
<td>optional</td>
<td>Assuming the corresponding object of the runs.tool.rules array has a message, this object will define which one of the messageStrings within the message property for the rule is used, and the values for the placeholders.</td>
</tr>
<tr>
<td>Message.id</td>
<td>mandatory</td>
<td>The id value corresponding to the messageString element that will be reported.</td>
</tr>
<tr>
<td>Message.arguments</td>
<td>mandatory</td>
<td>An array of values that correspond to the string placeholders that match the message.id.</td>
</tr>
<tr>
<td>locations</td>
<td>mandatory</td>
<td>It should include a value that corresponds to the physical (e.g., file, URI) or logical (e.g., fully qualified function name) location of the element where the issue was found. The actual implementation of the location may change for each tool, but it is typically described as a URI (file://... etc.), and may include a region (e.g., starting row/column, end row/column).</td>
</tr>
</tbody>
</table>

An example output of the SARIF runs.results property is presented below.

```json
"results": [
  {
    "ruleId": "sqlqvc/mssql/certificates-detection",
    "ruleIndex": 0,
    "message": {
      "id": "Default",
      "arguments": [
        "rsa",
        "cert_pvk_mk",
        "Private Key protected by master Key",
        "3072",
        "2/17/2024 5:15:37 AM"
      ],
      "locations": [
        {
          "physicalLocation": {
            "artifactLocation": {
              "uri": "sys.certificates",
              "uriBaseId": "REPO_ROOT",
            }
          }
        }
      ]
    }
  }
]
```
Recommended Message Property Placeholder Elements

This section describes the recommended elements that should be included as part of the runs.tool.rules.message placeholders to describe the elements found by the tool.

The custom elements listed in Table 10 allow upstream tools and aggregating tools to find and present these elements with ease within the SARIF log. For standardization purposes, we recommend publicly documenting the elements included in your tool and rules version. Table 11 adds recommended values to use when communicating the discovered vulnerable algorithm.

<table>
<thead>
<tr>
<th>Index</th>
<th>Element Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Name for the quantum-vulnerable algorithm detected</td>
<td>The textual representation of the algorithm object identifier being detected as at risk. For example: id-ecPublicKey or rsaEncryption (as listed in Table 11)</td>
</tr>
<tr>
<td>1</td>
<td>A unique identifier for the particular element</td>
<td>A unique identifier for the detected element that can be used to detect its usage and dependencies elsewhere. For example, in the case of a certificate, it could be a fingerprint of the certificate, a key opaque identifier, or a URI where a key is stored in a vault. If no unique identifier is available, any hint to find the element at risk in a system would be an acceptable alternative.</td>
</tr>
<tr>
<td>2+</td>
<td>Other</td>
<td>Starting on index 2, it may be useful to include data that may provide additional information on how the finding may have been used (e.g., what data it may be signing or protecting), or other elements that may assist (X.509 properties, other pieces of metadata, etc.).</td>
</tr>
</tbody>
</table>

Table 11 Vulnerable Algorithm Identifiers

<table>
<thead>
<tr>
<th>Algorithm Friendly Name</th>
<th>Algorithm Textual Representation</th>
<th>Algorithm Object Identifier</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA Keys</td>
<td>rsaEncryption</td>
<td>1.2.840.113549.1.1.1</td>
<td>RFC 3279 [36]</td>
</tr>
<tr>
<td>DSA Signature Keys</td>
<td>id-dsa</td>
<td>1.2.840.10040.4.1</td>
<td>RFC 3279</td>
</tr>
<tr>
<td>Diffie-Hellman Key Exchange Keys</td>
<td>dhpublicnumber</td>
<td>1.2.840.10046.2.1</td>
<td>RFC 3279</td>
</tr>
<tr>
<td>ECDSA and ECDH Keys</td>
<td>id-ecPublicKey</td>
<td>1.2.840.10045.2.1</td>
<td>RFC 3279</td>
</tr>
<tr>
<td>RSASSA-PSS Public Keys</td>
<td>id-RSASSA-PSS</td>
<td>1.2.840.113549.1.1.10</td>
<td>RFC 4055 [37]</td>
</tr>
</tbody>
</table>
### 5 Technologies

#### 5.1 Consortium Members

##### 5.1.1 Cisco

Cisco Systems, or Cisco, delivers collaboration, enterprise, and industrial networking and security solutions. The company’s cybersecurity team, Cisco Secure, is one of the largest cloud and network security providers in the world. Cisco’s Talos Intelligence Group, the largest commercial threat intelligence team in the world, is comprised of world-class threat researchers, analysts, and engineers, and supported by unrivaled telemetry and sophisticated systems. The group feeds rapid and actionable threat intelligence to Cisco customers, products, and services to help identify new threats quickly and defend against them. Cisco solutions are built to work together and integrate into your environment, using the “network as a sensor” and “network as an enforcer” approach to both make your team more efficient and keep your enterprise secure.

##### 5.1.2 IBM

IBM researchers, with their academic and industry partners, developed three of the four post-quantum cryptographic algorithms to be standardized by NIST. IBM z16 enterprise server leverages hybrid key agreement schemes and dual signing schemes to protect its infrastructure, and relevant to the project it provides a hardware security module and software libraries which allow its clients to experiment with FIPS 203 (CRYSTALS Kyber) and FIPS 204 (CRYSTALS Dilithium), two of the primary post-quantum algorithms slated to be standardized.

---

<table>
<thead>
<tr>
<th>Algorithm Friendly Name</th>
<th>Algorithm Textual Representation</th>
<th>Algorithm Object Identifier</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic Curve Cryptography Public Key Algorithm Identifiers</td>
<td>id-ecPublicKey</td>
<td>1.2.840.10045.2.1</td>
<td>RFC 5480 [38]</td>
</tr>
<tr>
<td>Restricted Algorithm Identifiers and Parameters</td>
<td>id-ecDH</td>
<td>1.3.132.1.12</td>
<td>RFC 5480</td>
</tr>
<tr>
<td>Restricted Algorithm Identifiers and Parameters</td>
<td>id-ecMQV</td>
<td>1.3.132.1.13</td>
<td>RFC 5480</td>
</tr>
<tr>
<td>Curve25519 and Curve448 Algorithm Identifiers</td>
<td>id-Ed25519</td>
<td>1.3.101.112</td>
<td>RFC 8410 [39]</td>
</tr>
<tr>
<td>Curve25519 and Curve448 Algorithm Identifiers</td>
<td>id-Ed448</td>
<td>1.3.101.113</td>
<td>RFC 8410</td>
</tr>
<tr>
<td>Curve25519 and Curve448 Algorithm Identifiers</td>
<td>id-X25519</td>
<td>1.3.101.110</td>
<td>RFC 8410</td>
</tr>
<tr>
<td>Curve25519 and Curve448 Algorithm Identifiers</td>
<td>id-X448</td>
<td>1.3.101.111</td>
<td>RFC 8410</td>
</tr>
</tbody>
</table>
IBM’s technology contribution consists of remote access to a z/OS image of an IBM z16 Mainframe environment, a workstation server, and the software, hardware, and tools needed to demonstrate the crypto discovery capabilities in support of the discovery workstream. This system is hosted at an IBM facility with remote access from the NCCoE Laboratory over a VPN, allowing collaborators to experiment with and understand IBM technologies that discover cryptographic usage within applications (with or without source code) and network connections.

5.1.3 Infosec Global (ISG)
ISG is a fast-growing cybersecurity company providing innovative solutions in the field of cryptographic agility management, cryptographic discovery, and post-quantum cryptography. ISG has a global footprint with offices in Canada, Switzerland, and the U.S. The ISG team combines the best cryptography experts (including the ‘father’ of SSL) with seasoned business leaders experienced in building and growing new businesses globally.

5.1.4 ISARA Corporation
ISARA is a security solutions company specializing in cryptographic risk management, quantum-safe cryptography, and cryptographic agility for today’s information technology ecosystems. Co-founded in 2015 by former BlackBerry security executives, ISARA’s cutting-edge technologies are enabling next-generation security for enterprises and governments. With ISARA, you can inventory and manage your cryptographic risks, future-proof your mission-critical systems, and achieve your quantum-safe and zero-trust goals. With an emphasis on interoperability, ISARA proudly collaborates on international standards-setting efforts.

5.1.5 Keyfactor
Keyfactor brings digital trust to the hyper-connected world with identity, encryption, and authentication for every machine, workload, human, and connected thing. By modernizing PKI, discovering and automating every digital certificate, and protecting critical software and product supply chains with secure digital signing and cryptography, Keyfactor helps organizations establish digital trust – then maintain it.

Keyfactor is committed to making quantum-ready PKI, signing, and cryptography solutions accessible for all, founding and actively supporting widely adopted open-source projects, including EJBCA, SignServer, and the FIPS 140-validated Bouncy Castle Cryptography APIs. As a member of the X9 Committee, Keyfactor is an active participant in standards-setting and has incorporated PQC algorithms into the Bouncy Castle APIs and its commercial PKI and signing solutions. With quantum-ready solutions and expertise, Keyfactor is working with customers to stay resilient in the post-quantum world.

5.1.6 Microsoft
Microsoft is committed to providing secure and trustworthy products and services to its customers. As such, Microsoft has been investing in PQC research, development, experimentation, and collaboration since 2014, playing a role in the emergence of PQC and public standards. In particular, Microsoft submitted four algorithms in NIST’s standardization effort. Microsoft is proud to participate in the Open Quantum Safe project, where they help develop the liboqs library used in this project and by many PQC
industry vendors. Microsoft established the Quantum Safe Program, aiming to accelerate and advance all quantum-safe efforts across the company from both technical and business perspectives.

5.1.7 SafeLogic

SafeLogic is a premier provider of cryptographic solutions that enable enduring privacy and trust in the ever-changing digital world. Founded in 2012, SafeLogic supplies organizations with strong, FIPS 140 validated cryptography via its FIPS Validation-as-a-Service. SafeLogic’s CryptoComply FIPS 140 validated cryptographic software modules support a broad range of platforms, programming languages, operating systems, and open-source libraries. SafeLogic expedites delivery of FIPS 140 CMVP certificates for its CryptoComply customers via RapidCert managed service. It then keeps those certificates active over time via MaintainCert, the company’s white glove managed service that uniquely provides both software support and certificate maintenance.

SafeLogic is keenly interested in building and supporting next-generation cryptographic solutions that will remain secure and compliant with the advent of quantum computers and will be easy for organizations to adopt. As part of that endeavor, SafeLogic is committed to working with NIST and fellow collaborators to capture lessons learned and develop best practices for adopting post-quantum cryptography.

5.1.8 Samsung SDS

Samsung SDS provides cloud and digital logistics services. Samsung SDS builds optimized cloud environments with Samsung Cloud Platform and provides all-in-one management service, as well as SaaS solutions proven successful in many use cases. One of the core capabilities for delivering their service is cybersecurity, and cryptographic technology plays a fundamental role to enhance security. To this end, Samsung SDS is engaged in various cryptographic research and development activities, including the design, implementation, and architecting of cryptographic techniques, including post-quantum cryptography.

5.1.9 SandboxAQ

SandboxAQ recently launched its Security Suite, a product for modern cryptography management which enables customers to pilot this change process towards a world where cryptography is observable, controllable, compliant, and secure. The SandboxAQ Security Suite is an end-to-end, crypto-agility platform that provides a full inventory of existing cryptography use, including vulnerability and compliance analysis, as well as a path to centrally managed, robust and agile cryptography. The result is protection from today’s attacks, as well as security against the future threat of a large-scale quantum computer.

5.1.10 wolfSSL

wolfSSL focuses on providing lightweight and embedded security solutions with an emphasis on speed, size, portability, features, and standards compliance. With its SSL/TLS products and crypto library, wolfSSL is supporting high-security designs in automotive, avionics, and other industries. In avionics, wolfSSL supports Radio Technical Commission for Aeronautics Software Considerations in Airborne Systems and Equipment Certification. In automotive, wolfSSL supports MISRA-C capabilities. For
government consumers, wolfSSL has a valid FIPS 140-2 certificate. wolfSSL supports industry standards up to the current TLS 1.3 and DTLS 1.3, offers a simple API and an OpenSSL compatibility layer, is backed by the wolfCrypt cryptography library, and provides 24x7 support and much more. wolfSSL’s products are open source, giving customers the ability to examine them.

5.2 Products and Technologies

The organizations listed in Section 5.1 have contributed the products and technologies listed in Table 12. For each product or technology, the table specifies the type of component, its name, and the function the technology will serve in the demonstration.

Table 12 Products and Technologies

<table>
<thead>
<tr>
<th>Component Type</th>
<th>Product or Technology Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerable Algorithm Discovery Platform System</td>
<td>IBM Z16 Mainframe</td>
<td>Computing platform supporting tools, applications, and crypto hardware.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform System</td>
<td>IBM Workstation Server</td>
<td>Computing platform supporting tools.</td>
</tr>
<tr>
<td>Quantum-Ready Crypto Implementation</td>
<td>IBM Crypto Express 8S</td>
<td>HSM providing support for cryptographic algorithm services.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>IBM z/OS Integrated Cryptographic Service Facility (ICSF)</td>
<td>Captures crypto-related information by writing System Management Facility (SMF) activity log records aggregating usage of cryptographic engines, services, and algorithms. The information is captured dynamically while the application is running. This facility can be used both to provide an inventory and to audit and observe during the complete migration process.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>IBM Application Discovery and Delivery Intelligence (ADDI) v.6.1</td>
<td>Serves as a static analysis tool that analyzes COBOL application source files capturing all ICSF crypto services, the parameters associated with the services, and valuable metadata. Produces a crypto discovery report.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>IBM Crypto Analytics Tool (CAT)</td>
<td>Provides snapshots of the z/OS environment by extracting security and cryptographic information based on configurable policies. Provides details on keys managed by ICSF and RACF. Helps identify insecure keys, algorithms, and enabled services.</td>
</tr>
<tr>
<td>Component Type</td>
<td>Product or Technology Name</td>
<td>Function</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>IBM z/OS Encryption Readiness Technology (zERT)</td>
<td>Collects and reports the cryptographic security attributes of Ipv4 and Ipv6 connections that are protected using the TLS/SSL, SSH, and lpsic cryptographic network security protocols. This tool helps provide the context for linking keys, certificates, and the applications using them. It identifies security protocols, crypto algorithms, key lengths, etc., all important information to have during the crypto discovery process.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>(SandboxAQ) Security Suite – Discovery Modules</td>
<td>Provides cryptographic observability capabilities. Analyzes IT infrastructure and creates a cryptographic inventory that allows stakeholders to monitor who/what, where, when, and how cryptography is used across an organization, including PQC.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>Samsung SDS Crypto Agility Platform for Enterprise (S-CAPE)</td>
<td>A platform that enables the discovery of PQ-vulnerable algorithms across the enterprise’s DevSecOps pipeline. By consolidating the data from various sensors, it provides visibility and estimated risk scores for the identified PQ vulnerabilities.</td>
</tr>
<tr>
<td>Cryptographic Inventory</td>
<td>InfoSec AgileSec Analytics Enterprise Server</td>
<td>AgileSec Analytics is an enterprise-grade security solution designed to enable companies in building a comprehensive and centralized inventory of all cryptographic assets, including cryptographic keys, keystores, X.509 certificates, cryptographic libraries, cryptographic algorithms, and cryptographic protocols deployed across their digital footprint.</td>
</tr>
<tr>
<td>Cryptographic Risk Assessment</td>
<td>InfoSec AgileSec Analytics Dashboard</td>
<td>AgileSec Analytics Dashboard is a core component of AgileSec Analytics enabling companies to review the cryptographic inventory and proactively identify cryptographic weaknesses, compliance gaps, or quantum-vulnerable objects based on a customizable cryptographic policy.</td>
</tr>
<tr>
<td>Cryptographic Data Collection</td>
<td>InfoSec AgileSec Sensors</td>
<td>AgileSec Analytics Sensors are core components of AgileSec Analytics and are used to scan different technologies and systems deployed within a digital footprint. The sensors can be used to scan hosts (filesystem, binary data, running processes, certificate store), network interfaces, CI/CD pipelines, application repositories, key management systems, PKI systems, HSM systems, and other technologies.</td>
</tr>
<tr>
<td>Cryptographic Vulnerability Remediation</td>
<td>AgileSec Analytics Vulnerability Response Connector</td>
<td>AgileSec Analytics Vulnerability Response Connector enables companies to seamlessly process the remediation of cryptographic vulnerabilities detected across their digital footprint with their existing solutions (e.g., ServiceNow).</td>
</tr>
<tr>
<td>Component Type</td>
<td>Product or Technology Name</td>
<td>Function</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cryptographic Agility</td>
<td>AgileSec Agility SDK</td>
<td>AgileSec Agility SDK enables companies building applications to natively integrate cryptographic agility within their sensitive business applications. With AgileSec Agility SDK, the cryptographic operations are abstracted from the developers and managed through policy, enabling seamless migration from classical to post-quantum, national, or any other future cryptographic standards.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>ISARA Advance® Cryptographic Discovery and Risk Assessment Tool</td>
<td>A platform for analyzing and inventorying the use of cryptography on enterprise networks. Includes features for inventorying devices and servers, and analyzing cryptographic risk to help you prioritize assets for cryptographic migrations. Results can be viewed in the platform’s easy-to-use dashboard or exported to other systems.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>(Samsung SDS) SECUI BLUEMAX NGF VE</td>
<td>A virtualized firewall configured to detect the presence of post-quantum vulnerable traffic.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>Microsoft CodeQL</td>
<td>Extension of Visual Studio Code and GitHub action that detects post-quantum vulnerable code.</td>
</tr>
<tr>
<td>Vulnerable Algorithm Discovery Platform</td>
<td>Cisco Mercury</td>
<td>Post-quantum vulnerable network packet metadata capture and analysis.</td>
</tr>
<tr>
<td>TLS Cryptographic Library Protocol Implementation</td>
<td>wolfSSL</td>
<td>A software library that implements TLS and DTLS 1.3, supporting classical and quantum-safe symmetric and asymmetric ciphers to be standardized by NIST.</td>
</tr>
<tr>
<td>Cryptographic Library Implementation</td>
<td>Keyfactor (Bouncy Castle)</td>
<td>(In partnership with the Legion of the Bouncy Castle Inc.) The Bouncy Castle libraries now include support for both classical and quantum-safe algorithms (NIST upcoming standards included), together with support for protocols such as TLS/DTLS, CMS, Time-Stamp Protocol, OpenPGP, and a variety of protocols around X.509 certificate generation and management.</td>
</tr>
<tr>
<td>Certificate Discovery and Management</td>
<td>Keyfactor Command</td>
<td>Command is a certificate discovery and lifecycle automation solution that provides centralized visibility, governance, and lifecycle automation for digital certificates at scale, helping organizations to identify and replace weak and non-compliant certificates.</td>
</tr>
<tr>
<td>Component Type</td>
<td>Product or Technology Name</td>
<td>Function</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Quantum-Ready PKI Platform</td>
<td>Keyfactor EJBCA</td>
<td>EJBCA is a modern, crypto-agile PKI platform that can be easily deployed in the cloud, on-premises, or in a hybrid mode. EJBCA supports issuance of certificates using both classical and quantum-safe algorithms.</td>
</tr>
<tr>
<td>Quantum-Ready Digital Signing</td>
<td>Keyfactor SignServer</td>
<td>SignServer is a flexible, crypto-agile signing solution that enables application, manufacturing, and product teams to digitally sign code and artifacts with quantum-safe algorithms.</td>
</tr>
</tbody>
</table>
6 Future Project Considerations

This volume presents a desired end state of an architecture which demonstrates tools that will discover vulnerable algorithms in an existing enterprise operational environment. We hope that publishing this document prior to integration activities will allow for refinements that will be submitted during the public comment period. As of the writing of this document, the project has installed, configured, and deployed technologies and tools from multiple consortium members that address discovery in our lab environment. These technologies are at the leading edge of the migration process, and as such, the project has taken a deliberate approach in learning the capabilities each platform provides by developing the demonstration plan described in Appendix C and O. The demonstration plan serves as a rough guide to develop the practical and readily adoptable use cases described within this guide. In future iterations we will finalize a data normalization scheme and add components that will enable automation and prioritization using a risk-managed approach.

Cryptographic discovery tools may produce inventories that indicate an organization is using more quantum-vulnerable cryptography than was expected, highlighting to the organization that they need to use a risk-based approach to mitigating the risks of the use of quantum-vulnerable cryptography. Future demonstrations within this project may address one or more of the risk methodologies identified in Section 3.4.3 to support the risk-based decisions an organization needs to make to prioritize its actions to migrate to post-quantum cryptography.

In the next revision of the draft document, we may broaden the scope of the project to include other discovery cases. One area that we are exploring is data-at-rest protection. When thinking of data-at-rest, the focus is primarily on the encryption algorithm, but attention must also be paid to the integrity of data-at-rest. An example is ensuring the integrity of audit records. In this case, hashes and/or digital signatures are often used to ensure that records have not been tampered with. Digital signatures will be susceptible to attack and broken by Shor’s algorithms. Example scenarios of data-at-rest protection could include database protection schemes, Secure Multipurpose Internet Mail Extensions (S/MIME), code signing, and record integrity.

Finally, the project has also submitted the CWE described in Section 3.4.2 to the Federally Funded Research and Development Center (The MITRE Corporation) that maintains the CWE program for review. In the next version of this document, we hope that the submission spurs conversation in this space, and that a consensus is achieved on one or more CWEs that will benefit organizations and industry alike.
## Appendix A  List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDI</td>
<td>IBM Application Discovery and Delivery Intelligence</td>
</tr>
<tr>
<td>AitM</td>
<td>Adversary-in-the-Middle</td>
</tr>
<tr>
<td>ALPN</td>
<td>TLS Application Layer Protocol Negotiation</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>CAI</td>
<td>Cryptographic Agility Index</td>
</tr>
<tr>
<td>CARAF</td>
<td>Crypto Agility Risk Assessment Framework</td>
</tr>
<tr>
<td>CAT</td>
<td>IBM Crypto Analytics Tool</td>
</tr>
<tr>
<td>CBOM</td>
<td>Cryptography Bill of Materials</td>
</tr>
<tr>
<td>CFDIR</td>
<td>Canadian Forum for Digital Infrastructure Resilience</td>
</tr>
<tr>
<td>CI/CD</td>
<td>Continuous Integration/Continuous Delivery</td>
</tr>
<tr>
<td>CISA</td>
<td>Cybersecurity &amp; Infrastructure Security Agency</td>
</tr>
<tr>
<td>CISO</td>
<td>Chief Information Security Officer</td>
</tr>
<tr>
<td>CPE</td>
<td>Common Platform Enumeration</td>
</tr>
<tr>
<td>CPS</td>
<td>Cyber Physical System</td>
</tr>
<tr>
<td>CRADA</td>
<td>Cooperative Research and Development Agreement</td>
</tr>
<tr>
<td>CRQC</td>
<td>Cryptanalytically Relevant Quantum Computer</td>
</tr>
<tr>
<td>CSF</td>
<td>Cybersecurity Framework</td>
</tr>
<tr>
<td>CWE</td>
<td>Common Weakness Enumeration</td>
</tr>
<tr>
<td>DAST</td>
<td>Dynamic Application Security Testing</td>
</tr>
<tr>
<td>DH</td>
<td>Diffie Hellman</td>
</tr>
<tr>
<td>DHS</td>
<td>U.S. Department of Homeland Security</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DoH</td>
<td>DNS over HTTPS</td>
</tr>
<tr>
<td>DSS</td>
<td>Digital Signature Standard</td>
</tr>
<tr>
<td>DTLS</td>
<td>Datagram Transport Layer Security</td>
</tr>
<tr>
<td>ECDH</td>
<td>Elliptic Curve Diffie Hellman</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Elliptic Curve Digital Signature Algorithm</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>EdDSA</td>
<td>Edwards-curve Digital Signature Algorithm</td>
</tr>
<tr>
<td>EDR</td>
<td>Endpoint Detection and Response</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract, Transform, and Load</td>
</tr>
<tr>
<td>FIPS</td>
<td>Federal Information Processing Standard</td>
</tr>
<tr>
<td>FS-ISAC</td>
<td>Financial Services Information Sharing and Analysis Center</td>
</tr>
<tr>
<td>HSM</td>
<td>Hardware Security Module</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority</td>
</tr>
<tr>
<td>ICSF</td>
<td>IBM z/OS Integrated Cryptographic Service Facility</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IKEv2</td>
<td>Internet Key Exchange Version 2</td>
</tr>
<tr>
<td>IMAPS</td>
<td>Internet Message Access Protocol Secure</td>
</tr>
<tr>
<td>IPsec</td>
<td>Internet Protocol Security</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LLMNR</td>
<td>Link-Local Multicast Name Resolution</td>
</tr>
<tr>
<td>ML-DSA</td>
<td>Module Learning with Errors Digital Signature Algorithm</td>
</tr>
<tr>
<td>MLWE-KEM</td>
<td>Module Learning with errors Key Encapsulation Mechanism</td>
</tr>
<tr>
<td>MQV</td>
<td>Menezes Qu Vanstone</td>
</tr>
<tr>
<td>NCCoE</td>
<td>National Cybersecurity Center of Excellence</td>
</tr>
<tr>
<td>NCF</td>
<td>National Critical Function</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NSA</td>
<td>National Security Agency</td>
</tr>
<tr>
<td>NSM</td>
<td>National Security Memorandum</td>
</tr>
<tr>
<td>OMB</td>
<td>Office of Management and Budget</td>
</tr>
<tr>
<td>PASTA</td>
<td>Process of Attack Simulation and Threat Analysis</td>
</tr>
<tr>
<td>PKCS</td>
<td>Public-Key Cryptography Standard</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PKIX</td>
<td>Public Key Infrastructure (X.509)</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>POP3S</td>
<td>Post Office Protocol 3 Secure</td>
</tr>
<tr>
<td>PQ</td>
<td>Post-Quantum</td>
</tr>
<tr>
<td>PQC</td>
<td>Post-Quantum Cryptography</td>
</tr>
<tr>
<td>QRWG</td>
<td>Quantum-Readiness Working Group</td>
</tr>
<tr>
<td>RACF</td>
<td>(IBM) Resource Access Control Facility</td>
</tr>
<tr>
<td>RDP</td>
<td>(Microsoft) Remote Desktop Protocol</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comment</td>
</tr>
<tr>
<td>S-CAPE</td>
<td>Samsung SDS Crypto Agility Platform for Enterprise</td>
</tr>
<tr>
<td>SARIF</td>
<td>Static Analysis Results Interchange Format</td>
</tr>
<tr>
<td>SAST</td>
<td>Static Application Security Testing</td>
</tr>
<tr>
<td>SBOM</td>
<td>Software Bill of Materials</td>
</tr>
<tr>
<td>SCA</td>
<td>Software Composition Analysis</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>SIEM</td>
<td>Security Information and Event Management</td>
</tr>
<tr>
<td>SLH-DAA</td>
<td>Stateless Hash-based Digital Signature Algorithm</td>
</tr>
<tr>
<td>SMF</td>
<td>System Management Facility</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SP</td>
<td>Special Publication</td>
</tr>
<tr>
<td>SSDF</td>
<td>Secure Software Development Framework</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>zERT</td>
<td>IBM z/OS Encryption Readiness Technology</td>
</tr>
</tbody>
</table>
Appendix B References


[16] MITRE (2023) ATT&CK. Available at https://attack.mitre.org/tactics/TA0009/


https://www.rand.org/pubs/research_reports/RRA1367-6.html


[27] Chandramouli R (2022) Implementation of DevSecOps for a Microservices-based Application with Service Mesh. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-204C. https://doi.org/10.6028/NIST.SP.800-204C


[35] OASIS Static Analysis Results Interchange Format Technical Committee (2023) Static Analysis Results Interchange Format (SARIF) Version 2.1.0 Plus Errata 01. Available at https://docs.oasis-open.org/sarif/sarif/v2.1.0/sarif-v2.1.0.pdf


Appendix C  Discovery Platform Lab Functional Demonstration Plan

C.1  Use Case 1

Scenario
Validate the discovery of quantum-vulnerable algorithms in TLS protocol version 1.2.

Steps
1. Configure discovery tool to scan layer 4 (transport) traffic. This process will depend on the individual discovery tool deployment methodology.
2. Modify test data packet capture IP addresses as needed for target network.
3. Replay TLS 1.2 test data onto network segment with deployed discovery tool using a utility such as tcpreplay.

Test Data
Encrypted Web traffic dataset from network monitoring and host-based monitoring across a large campus network

Expected Results
Discovery tool detects the presence of vulnerable key exchange/agreement and authentication algorithms, with results captured in an output file and/or dashboard.

C.2  Use Case 2

Scenario
Validate the discovery of quantum vulnerable algorithms in SSH protocol version 2.

Steps
1. Configure discovery tool to scan layer 4 (transport) traffic. This process will depend on the individual discovery tool deployment methodology.
2. Modify test data packet capture IP addresses as needed for target network.
3. Replay SSH 2.0 test data onto network segment with deployed discovery tool using a utility such as tcpreplay.

Test Data
AZSecure Data provided by AZSecure Data and the University of Arizona Artificial Intelligence Lab.
**C.3  Use Case 3**

**Scenario**

Validate the discovery of quantum-vulnerable algorithms in non-executable files in Windows-based operational systems.

**Steps**

1. Create a fully updated Windows 10/11 virtual machine.
2. Install the discovery tool sensor component if applicable.
3. Confirm proper communication between the sensor component and back-end systems provided by the discovery tool.
4. Move dataset files to the local disk of the target virtual machine.
5. Trigger the discovery sensor to perform a scan

**Test Data**

Using algorithms listed in Table 3, the following datasets are project-specific artifacts generated by the project team:

- PKCS#12 keystores created via OpenSSL
- Java keystores created via keytool utility
- PKCS#1 keystores created via OpenSSL
- PKCS#8 keystores created via OpenSSL
- OpenSSH keystores created via ssh-keygen
- OpenPGP keystores created via gpg

**Expected Results**

Discovery tool sensor detects files from dataset, with results captured in an output file and/or dashboard.

**C.4  Use Case 4**

**Scenario**

Validate the discovery of quantum-vulnerable algorithms in non-executable files in a Linux-based operational system.
Steps

1. Create a fully updated Ubuntu 22.04 virtual machine.
2. Install the discovery tool sensor component if applicable.
3. Confirm proper communication between the sensor component and back-end systems provided by the discovery tool.
4. Move dataset files to the local disk of the target virtual machine.
5. Trigger the discovery sensor to perform a scan.

Test Data

See Use Case 3.

Expected Results

Discovery tool sensor detects files from dataset, with results captured in an output file and/or dashboard.

C.5 Use Case 5

Scenario

Validate the discovery of quantum-vulnerable algorithms in executable files on Windows-based operational systems that are representative of a typical enterprise deployment.

Steps

1. Install, configure, and deploy discovery tool endpoint agent.
2. Note discovered artifacts, given discovery platform capabilities.

Test Data

Windows 11-based client with the following software packages:

- Google Chrome
- Java Runtime Environment
- VLC Media Player
- OpenVPN Connect
- GRR Rapid Response
- Slack

Expected Results

The discovered artifacts are collected and optionally transmitted to a back-end system for review.
C.6 Use Case 6

Scenario

Validate the discovery of quantum-vulnerable algorithms in executable files on Linux-based operational systems that are representative of a typical enterprise deployment.

Steps

1. Install, configure, and deploy discovery tool endpoint agent.
2. Note discovered artifacts, given discovery platform capabilities.

Test Data

Ubuntu 22.04 Server Edition image with all base services enabled.

Expected Results

The discovered artifacts are collected and optionally transmitted to a back-end system for review.

C.7 Use Case 7

Scenario

Validate the discovery of quantum-vulnerable algorithms in code that leverages cryptography in a CI/CD pipeline (IDE plugin).

Steps

1. Create a development project in IDE using source code from dataset.
2. Trigger the discovery tool within the IDE.

Test Data

- Source Code
  - RSA Encryption/Description API usage
  - ECDSA Sign/Verify API usage
- Library/Process
  - KeyPairGenerator, KeyFactory, Signature, SignatureException Java class

Expected Results

Discovery tool identifies files and lines of code that use the methods described in the test dataset and displays results within the IDE.

C.8 Use Case 8

Scenario

Validate the discovery of quantum-vulnerable algorithms in code that leverages cryptography in a CI/CD pipeline (Repository).
Steps

1. Clone repository containing source code from dataset.
2. Make a change in the codebase (e.g., add a print statement such as “hello world”).
3. Commit the change.
4. Create a pull request in the configured remote repository.
5. Code scan is triggered automatically.

Test Data

- Source Code
  - RSA Encryption/Description API usage
  - ECDSA Sign/Verify API usage
- Library/Process
  - KeyPairGenerator, KeyFactory, Signature, SignatureException Java class

Expected Results

Discovery tool identifies files and lines of code that use the methods described in the test dataset and displays results within the repository console.
Appendix D  IBM Z16 Remote Discovery Platform Functional Demonstration Plan

D.1 Use Case 1

Scenario
Validate the discovery of post-quantum vulnerable algorithms in executable modules in the z/OS-based operational system.

Steps
1. Enable the ICSF crypto usage tracking feature of z/OS.
2. Run the job associated with the application to be evaluated.
3. Retrieve the SMF 82 and/or 113 records associated with that job from the dataset.
4. Run the supplied job to parse desired information from the SMF dataset.
5. Generate the desired reports from the collected data.

Test Data
- SMF 82 log records – ICSF and/or
- SMF 113 log records – Processor Activity Instrumentation

Expected Results
Discovery feature detects use of ICSF and CPACF crypto, including engines, algorithms, and key lengths used, with results captured in SMF data sets.

D.2 Use Case 2

Scenario
Validate the discovery of post-quantum vulnerable algorithms in code that leverages ICSF cryptography in COBOL application source.

Steps
1. Download code to Application Discovery and Delivery Intelligence system.
2. Start ADDI.
3. Enable the JSON configuration file.
4. Select the source code to analyze.
5. Perform the ADDI build step.
6. Review the report.

Test Data
COBOL source code containing:
- ICSF crypto calls (CSNBENC, CSNDDSG, CSNDPKE, CSNDEDH, etc.)
- Rule array keywords
Expected Results

Discovery tool identifies and reports cryptographic calls, important parameters, call line numbers, and metadata, and writes it to a file for later display or export to a CSV file.

D.3 Use Case 3

Scenario

Validate the discovery of post-quantum vulnerable algorithms in code that leverages a crypto API in COBOL application source.

Steps

1. Download code to Application Discovery and Delivery Intelligence system.
2. Update the JSON file to identify the crypto calls you would like to search for.
3. Start ADDI.
4. Enable the custom configuration file.
5. Select the source code to analyze.
6. Perform the ADDI build step.
7. Review the report.

Test Data

COBOL source code containing:
- Crypto calls (e.g., c_sign, c_verify) as defined in JSON configuration file
- Parameters

Expected Results

Discovery tool identifies and reports cryptographic calls, important parameters, call line numbers, and metadata, and writes it to a file for later display or export to a CSV file.

D.4 Use Case 4

Scenario

Validate the discovery of post-quantum vulnerable algorithms in code that leverages cryptography in a CI/CD pipeline (IDE plugin).

Steps

1. In an IDE, populate the data.
2. Check-in and commit in git.
3. Execute pipeline, which triggers ADDI’s build.
4. Perform the discovery.

Test Data

Source code with crypto calls such as:
- RSA Encryption/Decryption API usage
D.5 Use Case 5

Scenario

Validate the discovery of post-quantum vulnerable algorithms in network traffic.

Steps

1. Capture a snapshot of network traffic by collecting SMF 119 records.
2. Make SMF data available to z/OS Encryption Readiness Technology.
3. Log on to z/OS Management Facility.
4. Start the z/Network Analyzer.
5. Select the run and manage query option.
6. Create a query that looks for ECC/DH/DSA and other quantum-vulnerable algorithm usage.
7. Run query to generate a report.

Test Data

SMF 119 Data:

1. Cipher suites
2. TLS version
3. Certificate information – expiration, serial numbers, etc.
4. IP address
5. Protocol session identifiers
6. Protection information – key lengths, algorithm, etc.
7. User ID, ports, jobnames

Expected Results

Discovery tool can display on a dashboard and/or produce a report that identifies connections that are using vulnerable cryptography.

D.6 Use Case 6

Scenario

Discover if quantum-vulnerable keys are managed by ICSF or RACF.

Steps

1. Start the Crypto Analytics Tool.
2. Take a snapshot of the certificates and keys and their related protection/authorization information stored in the RACF data base and the keys in the ICSF key data sets (CKDS, PKDS, and TKDS).
3. Define a policy to identify the characteristics of the keys and certificates to be evaluated.
4. Apply the policy, which will create a report that identifies keys and certificates matching the criteria specified in the policy.
5. Review the results.

Test Data
ICSF key data sets and RACF information

Expected Results
The discovery tool:
- Identifies keys that are considered quantum-vulnerable
- Identifies keys that match the desired search criteria
- Identifies characteristics of certificates managed by RACF
- Identifies callable services that are available for use in the HSM that should be disabled
- Displays information on the dashboard. Reports can also be generated.