NIST SPECIAL PUBLICATION 1800-40B

Automation of the NIST Cryptographic Module Validation Program

Christopher Celi
Alex Calis
Murugiah Souppaya*
Computer Security Division
Information Technology Laboratory

William Barker
Domestic Guest Researcher
Information Technology laboratory

Karen Kent
Trusted Cyber Annex

Shawn Geddis
Katalyst

Raoul Gabiam
The MITRE Corporation

Stephan Mueller
Yi Mao
atsec information security

Barry Fussell
Andrew Karcher
Cisco

Douglas Boldt
Amazon Web Services

* Former employee; all work for this publication was done while at that organization.

April 2026

INITIAL PUBLIC DRAFT

This publication is available free of charge from
https://www.nccoe.nist.gov/automation-nist-cryptographic-module-validation-program

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.

While NIST and the NCCoE address goals of improving management of cybersecurity and privacy risk through outreach and application of standards and best practices, it is the stakeholder's responsibility to fully perform a risk assessment to include the current threat, vulnerabilities, likelihood of a compromise, and the impact should the threat be realized before adopting cybersecurity measures such as this recommendation.

National Institute of Standards and Technology Special Publication 1800-40, Natl. Inst. Stand. Technol. Spec. Publ. 1800-40B, 89 pages, (April 2026), CODEN: NSPUE2

FEEDBACK

You can view or download the guide at the NCCoE Automation of the NIST Cryptographic Module Validation Program project page.

With this initial public draft of the final report, NIST NCCoE is now asking for feedback where clarifications might be beneficial.

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

Public comment period: April 15, 2026 through June 1, 2026

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.

ABSTRACT

The Cryptographic Module Validation Program (CMVP) validates third-party assertions that cryptographic module implementations satisfy the requirements of Federal Information Processing Standards (FIPS) Publication 140-3, Security Requirements for Cryptographic Modules. Historically, the CMVP validation review process has struggled to keep pace with the volume of cryptographic modules and accelerated software release cycles, contributing to delays in validation timelines. The NIST National Cybersecurity Center of Excellence (NCCoE) has undertaken the Automated Cryptographic Module Validation Project (ACMVP) to explore how automation can improve the efficiency and timeliness of CMVP operations and processes. The project demonstrates how structured test evidence, standardized submission protocols, and supporting modernized computing infrastructure can streamline the submission and review of validation artifacts.

This publication describes the approaches and tools demonstrated by the ACMVP team. The publication describes the results of an ACMVP Test Evidence (TE) Workstream and Protocol Workstream, as demonstrated in a laboratory environment developed by the project's Research Infrastructure Workstream. The combined impact of these workstreams is intended to provide automation improvements to improve submission quality and enable a more efficient CMVP review process.

KEYWORDS

Automated Cryptographic Module Validation Project (ACMVP); Cryptographic Module Validation Program (CMVP); cryptography; cryptographic module; cryptographic module testing; cryptographic module validation.

ACKNOWLEDGMENTS

Collaborators participating in this project submitted their capabilities in response to an open call in the Federal Register for all sources of relevant security capabilities from academia and industry (vendors and integrators). The following respondents with relevant capabilities or product components signed a Cooperative Research and Development Agreement (CRADA) to collaborate with NIST in a consortium to build this example solution.

- Acumen Security
- AEGISOLVE
- Apple
- atsec information security
- AWS
- Cisco
- Katalyst
- Lightship Security
- Microsoft
- NXP Semiconductors
- SUSE

ACKNOWLEDGEMENTS

Contributors to each workstream are listed in the corresponding sections below. Additionally, the following people and organizations contributed to the project outside of a workstream: Courtney Maatta, Rochelle Casey, Alicia Squires, Margaret Salter, Tim Ness, Damian Zell, Derrick Williams III, Jeff Wright, Mickey Iqbal, and David Browning of Amazon; Dave Hawes, Gavin O'Brien, Tim Hall, Matt Scholl, Cherilyn Pascoe, Jim St. Pierre, Kevin Stine, Ann Rickerds, Shawn Winhoven, Jeffrey J McIntyre, Anil Das, Edgar Garay, Jim Simmons, Robert Staples, Rob Densock, and Blair Heiserman of NIST; Jason Arnold of HII; William Barker of Strativia LLC; Karen Scarfone of Scarfone Cybersecurity; and Heather Flanagan of Spherical Cow Consulting.

The project team recognizes and appreciates Apostol Vassilev of NIST for leading the project at the inception and formulating the three workstreams and associated activities.

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 “may 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 guidelines or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidelines 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 to hold 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 guidelines or requirements in this ITL draft publication, either:

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

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-testing@nist.gov

Contents

List of Figures

List of Tables

Executive Summary

The National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) jointly conduct the Cryptographic Module Validation Program (CMVP) to verify that cryptographic module implementations conform to the Federal Information Processing Standard (FIPS) 140-3. Federal agencies, many private sector companies, and foreign government organizations require cryptographic products to be CMVP-validated prior to being purchased and used. Since the CMVP's inception in 1995, the volume, complexity, and speed-to-market of cryptographic modules seeking validation have steadily increased. In the case of cryptographic software modules, product cycles now move in weeks or months. In recent years the testing and validation process could take up to 2 years to complete due to a combination of limited staff resources, incomplete submissions, and documentation ambiguities that extend the validation timeline. These delays limit product options for organizations required to use validated cryptography.

NIST is undertaking a broad effort to modernize and automate CMVP processes with the goal to reduce dependence on manual review and timeliness and scalability of the validation process. This guide presents the approach developed by the National Cybersecurity Center of Excellence (NCCoE) and its collaborators to demonstrate automation capabilities that support the modernization of CMVP, helping align cryptographic validation processes with modern software development and deployment environments.

The NCCoE Automated Cryptographic Module Validation Project (ACMVP) explored how structured test evidence, standardized submission protocols, and modern technology infrastructure across three coordinated workstreams enable automation of the validation processes:

1. Test evidence workstream: developed structured representation of FIPS 140-3 test evidence in order to enable machine-readable submissions. This was foundational in the creation of TE filters; listing required TEs based on the type of module being validated, reducing ambiguities and ensuring completeness and accuracy of each submission prior to CMVP review.
2. Protocol workstream: developed standardized submission interfaces to support automated checks prior to CMVP review. The developed schemas, protocol, and application reduce delays due to incompleteness and ambiguities in submissions by providing an interface that offers instant feedback prior to formal review by the CMVP. This ensures module submissions meet all requirements and follow a standardized format removing ambiguities. The standardized format also enables future work from the CMVP to automate the review of evidence.
3. Research infrastructure workstream: The CMVP develops and hosts applications for the purpose of performing validations and reviews. The supporting infrastructure was rehosted in the cloud with minimal architectural changes. The workstream developed a modern cloud architecture and supporting infrastructure/services to demonstrate efficiencies gained in transitioning to a cloud-native architecture.

The scope of the NCCoE ACMVP project focused on machine readable payloads to ensure the completeness, accuracy and consistency of validation submissions to automate the validation process.

The capabilities described in this guide are actively being integrated into the production CMVP environment in 2026 to support a faster time-to-market of validated cryptographic modules.

The primary goal of the guide is to describe the NCCoE-developed automation application, process, and protocols as they are being integrated into the NIST CMVP. These descriptions are intended to help cryptographic and security testing labs, technology producers, and validation authorities leverage this modern approach to shorten the validation cycle while maintaining and improving assurance levels.

A secondary goal of the guide is to provide practical information that can be applied to the CMVP infrastructure to further modernize the underlying systems and applications and support leveraging cloud-native services to maximize the efficiency of validation processes.

1 Overview

This publication summarizes key challenges faced by the Cryptographic Module Validation Program (CMVP) and presents an approach to demonstrate automation of the manual processes. It highlights the workstreams' development of protocols and structured data to streamline the processes and increase efficiency.

1.1 Challenge

The CMVP validates third-party assertions that cryptographic module implementations satisfy the requirements of Federal Information Processing Standards (FIPS) Publication 140-3, Security Requirements for Cryptographic Modules [1]. Under the CMVP, cryptographic modules undergo third-party testing by National Voluntary Laboratory Accreditation Program (NVLAP) accredited laboratories, and the processes and results are validated under a program run by the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS). Current industry cryptographic product development, production, and maintenance processes place significant emphasis on time-to-market efficiency. A number of elements of the validation process are manual in nature, and the period required for third-party testing and government validation of cryptographic modules is often incompatible with industry development and release cycles. Given the increasing velocity of product and software updates, as well as the migration towards post-quantum cryptography (PQC), there is a significant risk of continued backlog in validation without a shift toward automation.

1.2 Solution

The NIST National Cybersecurity Center of Excellence (NCCoE), in collaboration with the CMVP, has undertaken a project to demonstrate the value and practicality of automation support to improve the responsiveness of CMVP. The intent of the Automated Cryptographic Module Validation Project (ACMVP) is to support improvement in the efficiency and timeliness of CMVP [2]operations and processes. This NCCoE effort builds on other automation initiatives within the CMVP ecosystem, such as the successful completion of the automation of the Cryptographic Algorithm Validation Program (CAVP); the rollout of WebCryptik, an application for submitting test results to the CMVP; and the automation of entropy data testing evidence processing for the Entropy Source Validation (ESV) program. The initiative will provide mechanisms for the structural presentation of testing evidence by NVLAP-accredited parties to facilitate the automation of evidence validation by the CMVP.

The ACMVP's goal is to enable automated test report review where feasible for each of the test requirements found in FIPS 140-3 [2] and International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 24759 [3], which FIPS 140-3 incorporates by reference.

The module testing and reporting aspects of module validation, according to ISO/IEC 24759, combine functional and nonfunctional security requirements. This project aims to streamline the test methods for the functional testing of specific classes of technologies (e.g., software modules) and the corresponding reporting of functional and non-functional security requirements. The project is working to demonstrate a suite of tools to modernize and automate manual review processes.

The project was executed across three workstreams in collaboration with accredited test laboratories, vendors, and participating authorities. The Test Evidence (TE) Workstream focused on how individual requirements (as defined in ISO/IEC 24759 as Test Evidences) are classified by the lab and the validation authority. The Protocol Workstream has demonstrated a server and client implementation to accept and process module validation submissions. The Research Infrastructure Workstream developed the laboratory environment in which the server resides. The combined impact of these workstreams is resulting in automation improvements to the operations of the CMVP.

1.3 How to Use This Guide

This guide offers two content formats: the "High-Level Document in PDF Format" (this document) and implementation details on the ACMVP Documentation website to which this High-Level Document links. The PDF document serves as an introduction to the project, including a high-level summary of the project goals, ACMVP capabilities, demonstrated implementations, and project findings. The linked web pages provide in-depth details on NIST SP 1800-40: *Automation of the NIST Cryptographic Module Validation Program*, including full TE classification lists, protocol descriptions implemented by the server and clients, and the technologies leveraged, with specific integrations and configurations. Readers are encouraged to begin by reading the document in PDF format (this document) to gain high-level insight into the project. Readers may then drill down from this document into the deeper sections of the linked web pages to access in-depth information as needed.
Therefore, this document is organized as follows:

- Section 2 describes the approach taken to demonstrate the use of automation tools by the CMVP.
- Section 3 describes the work of, and capabilities demonstrated by, the Test Evidence Workstream.
- Section 4 describes the protocol and implementations demonstrated by the Protocol Workstream.
- Section 5 describes the laboratory development and demonstration environment developed by the Research Infrastructure Workstream.
- Section 6 concludes this document by sharing takeaways as recommended steps for continuing automation of CMVP processes.

Anyone interested primarily in the lessons learned from the project should focus on the takeaways provided in Section 6.

2 Approach

2.1 Audience

The primary audience for this report is technology, security, and privacy program managers and architects, software developers, engineers, and IT professionals, especially those involved with the CMVP, accredited cryptography and security testing labs, and conformance offices at companies that produce security software and hardware.

2.2 Scope

The project demonstrates tools that will modernize and automate manual review processes in support of existing CMVP policy. The scope of the project is focused on enabling an accredited lab to make a full module submission to the CMVP. This includes technical testing evidence reported to the CMVP and the construction of a security policy document that goes on the final module validation certificate. The automated tools use a server/client testing model, allowing testing labs to submit reports to the CMVP incrementally. Due to the server-based implementation acting on behalf of the CMVP, automated review can occur before a submission reaches the CMVP. The scope of the ACMVP project activities encompassed the following tasks to allow for automating module validation of a new module submission at any security level:

- a list of standard test methods for the functional testing of modules,
- a full reporting of applicable functional and non-functional security requirements,
- a cloud-based infrastructure to host the environment at NIST,
- and a protocol that can enable the generation and validation of standardized evidence produced by the operational testing of an Implementation Under Test (IUT).

While evaluating automation for a software module which was the initial project scope, it was found that enabling hardware modules for automation was not a significant addition. As a result, the ACMVP has focused on all modules (software and hardware) across security levels 1 through 4, with their first submissions to the CMVP. Future phases can help automate other submission types for the CMVP, such as submissions that address only Critical Vulnerability Enumerations (CVEs) or submissions to add operating environments to an existing validation.

2.3 Workstreams

The project was organized into three workstreams: the Test Evidence (TE) Workstream, the Protocol Workstream, and the Research Infrastructure Workstream.

The structured application of the TE classification and filtering proposed by the TE Workstream plays a crucial role in streamlining the validation process for cryptographic modules under FIPS 140-3. The TE classification categorizes requirements so that the developed proof of concept ACMVP server can require direct body of evidence. By leveraging both basic and supplemental filters, the evaluation process ensures that only relevant test evidence is considered, reducing redundancy while maintaining rigorous security standards. This approach enhances efficiency, supports automation, and enables a

more scalable validation framework. This shifts the applicability of requirements to a centralized community consensus rather than treating them as an additional item for the reviewer to consider.

The Protocol Workstream defines the interactions between the CMVP server and the ACMVP clients supporting a proof-of-concept of automation capabilities. The system is inspired by the Automated Cryptographic Validation Protocol (ACVP) and supports a full module submission to the CMVP. This includes describing the capabilities of the module, addressing how each requirement from FIPS 140-3 is met, and generating a security policy. It integrates with WebCryptik to provide a front-end for generating large JavaScript Object Notation (JSON) payloads and leverages the CMVP's internal security policy builder application to ensure consistent documentation.

The ACMVP project is cloud-based. The Research Infrastructure Workstream team adopted an iterative approach to modernize the CMVP supporting infrastructure to complement the developed proof of concept ACMVP server. Each iteration introduced progressively advanced architectures, leveraging cloud-native services to improve scalability, portability, deployment speed, and security of the modernized application. The modernization efforts have resulted in a containerized server application compatible with both Windows and Linux platforms. It integrates a managed database service to enhance operational efficiency and features a fully automated Continuous Integration/Continuous Deployment (CI/CD) pipeline to simplify and streamline deployments on a Linux platform. Authentication mechanisms have been modernized to incorporate cloud-native solutions, including the AWS Network Load Balancer (NLB).

The combined impact of these workstreams is automation and improvements to the CMVP's operations.

2.4 Assumptions

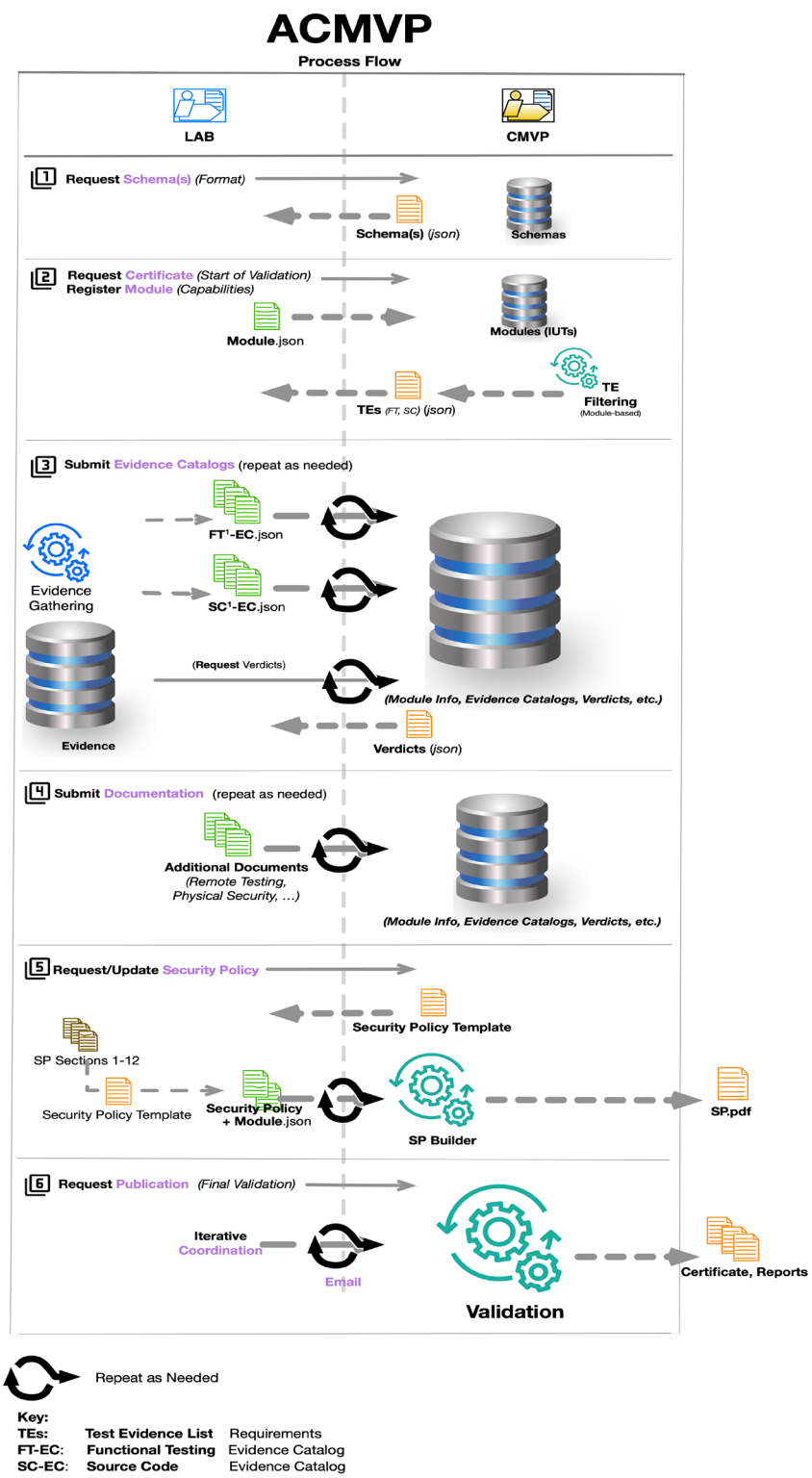
The CMVP handles many submission types for modules seeking validations or revalidations. Modules may wish to add algorithms, not impacting other previously validated functionality. Modules may wish to update a version to address a reported CVE. ACMVP acknowledges that there are many more submission types than a full module submission for initial validation. The project operates under the assumption that a full module submission will cover the techniques and tools needed to apply to other, more specific submission types. This extension can be handled by the CMVP or by an interested community such as the Cryptographic Module User Forum (CMUF).

An additional assumption made by this project is that fully automated module validation is not the immediate goal of the project. Many tasks performed manually by CMVP reviewers can and are automated by ACMVP. The goal of the project is to reduce the amount of manual work done, but not necessarily eliminate it entirely. A human reviewer will be needed to ensure a submission makes coherent sense, for example, a network switch is not submitted as a software module.

2.5 Workflow of an Automated CMVP

The ACMVP Process Workflow diagram in Figure 1 expresses the expected data exchange between an accredited Lab and the CMVP for the steps in the formal submission of the evidence, supporting documents, and request for validation. The left side of the diagram represents the work performed and the data exchanged by the Lab while the right side represents this workflow from the CMVP perspective. The center line represents the network boundary for the transmission and retrieval of the corresponding

JSON payloads between the Lab and CMVP. The workflow diagram is divided horizontally into phases of the ACMVP protocol as described by their individual subtitles. The following is a high-level description of each one of the well-defined phases of the workflow.



471 **Figure 1 Workflow of an automated CMVP**

2.5.1 Request Schema(s)

Simplifying the process and ensuring valid JSON payload submissions, ACMVP offers the authoritative JSON Schemas for each of the JSON payload submission types.

JSON Schemas currently available:

- Certificate Request Registration
- Functional Test Evidence Submission
- Other Documentation Evidence Submission
- Security Policy Submission (module.json)
- Source Code Evidence Submission

2.5.2 Request Start of Validation and Register Module (Capabilities)

A Module Validation workflow is initiated with a Module Certificate Request from the Lab to CMVP which includes the vendor's Module capabilities and uniquely identify the Implementation Under Test (IUT). In response, CMVP will provision a Certificate Request identifier and generate a scoped list of Test Evidence Identifiers (TEs) which requires Lab submitted rationale and captured evidence metadata.

2.5.3 Submit Evidence Catalogs

The heavy lifting begins with the Lab capturing all of the relevant evidence from Functional Testing, Source Code, and relevant vendor documentation. While capturing the evidence, the Lab submits selected metadata, referred to as evidence catalog, for each of the previously identified TEs identified by CMVP for validation of a module with the claimed capabilities. The module testing, evidence gathering, and evidence catalog submissions can be done as a complete package, a group of submissions, or even individually for each identified TE allowing for significant flexibility for the Lab. With each submission, CMVP will acknowledge receipt of the submission which should not be interpreted as validation results just yet.

2.5.4 Submit Additional Documentation

Module validations often require in review of various lab, vendor, and technology specific documentation (eg. Remote Testing, Physical Security, etc.) in order for CMVP to properly validate the Module has been properly tested and is fully conformant to the current cryptographic and security requirements. This phase enables Lab submission of all relevant and supportive documentation.

2.5.5 Request / Update Security Policy

Significantly simplifying the **Security Policy** document generation stems from the reduction to Lab submission of individual Security Policy sections with content to be inserted into a CMVP defined Security Policy template. This reduces the workload in preparation and submission while ensuring consistent content, look and coverage is guaranteed by CMVP's generation and publishing of the final Security Policy document according to their template. Reduces resource drain for everyone while ensuring consistency. As is enabled with the evidence catalog submissions, this phase enables iterative submissions to CMVP as information becomes available to the lab.

2.5.6 Request Publication

The final phase of a Module Validation workflow is the Request Publication. This phase triggers the formal Validation by CMVP involving an iterative Coordination with the Lab and finalization resulting in posting of Security Policy, and the Module Validation Certificate.

3 Test Evidence Workstream

The TE Workstream focused on how the individual requirements (known in ISO/IEC 24759 as Test Evidences) are classified by the lab and the validation authority. The TE Workstream defined a structured application of test evidence (TE) classification and filtering, which is crucial in streamlining the validation process for cryptographic modules under FIPS 140-3. This structure enabled TE classification, filtering on relevant module submission requirements, and the development of an application to streamline the validation process of submitted cryptographic modules.

3.1 TE Workstream Collaborators

The ACMVP TE Workstream (WS) was led by Yi Mao of atsec information security and Shawn Geddis of Katalyst under the NCCoE ACMVP leadership of Murugiah Souppaya and Christopher Celi of NIST. The workstream is indebted to the invaluable contributions of Alex Calis of NIST CMVP, who served as the co-chair of this WS for more than two years. The WS benefited from contributions from the atsec information security team, including but not limited to Stephan Mueller, Walker Riley, Swapneela Unkule, and Jeremy Wesevich; the Intertek Acumen Security team led by James Reardon with Chris Bell, Sowndar Gillan Gopi, and Rutwij Kulkarni; the AEGISOLVE team including but not limited to Travis Spann, Javier Martel, Mike McCarl, and Debbie Harrington; Ryan Thomas of Lightship Security; Barry Fussell and Andrew Karcher of Cisco; Alicia Squires and Courtney Maatta of Amazon; Marc Ireland of NXP; Mike Grimm of Microsoft; Ivan Teblin and Blaine Stone of SUSE; and Michael Dimond of the MITRE Corporation.

The main accomplishments of the TE WS are as follows:

- Classification/categorization of TEs
- A well-defined structure for test evidence data represented in JSON
- Alignment of the CMVP's Documentation TE List with TE classifications
- Recommended test methods for functional testing TEs
- TE filtering to make the report focus only on the relevant requirements

3.2 Test Evidence Classification

The TE WS has identified and sorted categories of test evidence required for CMVP validation that can readily be automated in a reporting format that is consistent with the current WebCryptik used by CMVP. The TE WS has also identified those test evidence classes for which manual processes are still needed.

3.2.1 TEs Requiring Vendor Documentation

The TE WS team has classified test evidence into the following categories, depending on what needs to be checked, inspected, or tested, and how the vendor evidence (VE) is supposed to be provided:

1. Assessments based on reviewing the vendor documentation, especially the SP
2. Assessments based on inspecting the module's source code
3. Assessments based on exercising/executing the module to cover:
    a. Functional testing

b. Physical security

Section 3 of this document covers items 1 and 2. Section 4 covers item 3a. Item 3b is out of scope of this project but may be part of future work due to the nature of physical security tests. For example, the ability to remove a tamper-resistant sticker without evidence of tampering.

The required documentation for a FIPS validation is specified in the NIST SP 800-140A [4], which modifies the vendor documentation requirements of ISO/IEC 19790 Annex A [5]. Hereafter, the vendor-documentation-dependent TEs will be indicated as **140A-TE**s. Those TEs require the tester to verify the presence and accuracy of information within the vendor documentation or verify statements based on information from the documentation.

The overall category of 140A-TEs, as opposed to the TEs depending on functional test (hereafter **FT-TE**s), is relatively clear. They are indicated by the keyword “verify” as in the following examples:

- “verify the name and version as indicated in AS04.13” (e.g., TE04.33.01)
- "verify the vendor documentation" (e.g., TE04.05.01)
- "verify that the vendor provided documentation" (e.g., TE05.05.01)
- "verify, by inspection and from the vendor documentation" (e.g., TE05.15.01)
- "verify the vendor documentation, and by inspection" (e.g., TE06.10.01), "verify by inspection, or from the vendor documentation" (e.g., TE07.15.01)
- "verify … as documented" (e.g., TE07.27.01)
- "verify … are documented" (e.g., TE07.33.01)
- "verify the vendor documentation shows … " (e.g., TE10.09.01)
- "verify … through the procedure documented in …" (e.g., TE10.11.01)

The 140A-TEs may or may not depend on the SP. They may depend on source code or other proprietary documentation. So, the **140A-TE**s can be further divided into three sub-categories:

- **SP-TE**s: TEs depend on the information provided by the public-facing Security Policy (SP). The NIST SP 800-140Br1 [6] is to be used in conjunction with ISO/IEC 19790 Annex B and ISO/IEC 24759 section 6.14. It also specifies the order of the SP. Ideally, Special Publication 800-140Br1 should require SP to include all information to satisfy the SP-dependent TEs.
- **SC-TE**s: TEs require source code review. It may not be intuitive that source code falls under vendor documentation. Source code review requires special care and attention. Therefore, we separate these SC-TEs from the TEs depending on other vendor documentation.
- **OD-TE**s: If a 140A-TE is not an SP-TE, nor an SC-TE, then we designate it as an OD-TE, meaning the TE depends on Other Documents such as Finite State Model (FSM), Component List (CL), design documents, user guidance, or configuration management manual. For each OD-TE, the lab is required to receive and verify specific documentation requirements. The CMVP only needs to receive assurance that the lab has completed each OD-TE task.

Here are some examples:

- SP-TEs: 140B requires SP to provide the information
  - TE04.47.01:

The tester shall verify that the security functions used to authenticate operators are all approved security functions.

- TE04.48.01:
  The tester shall verify that the authentication mechanism used to authenticate operators is an approved one.

- SC-TEs: TEs that depend on source code inspection
  - TE03.07.05:
    The tester shall verify that the vendor documentation specifies how the cryptographic module ensures that all data output via the data output interface is to be inhibited during error states or self-test conditions. The tester shall also verify, by inspection of the design of the cryptographic module, that the data output interface is, in fact, logically or physically inhibited under these conditions.
  - TE03.15.05:
    The tester shall examine the applicable source code(s) to ensure that the identified component is actually validating the documented format.
- OD-TEs: requires a rationale of correctness, FSM, or SW/FW CL
  - TE03.19.03:
    The tester shall verify the correctness of any rationale provided by the vendor. The burden of proof is on the vendor; if there is any uncertainty or ambiguity, the tester shall require the vendor to produce additional information as needed.
  - TE11.08.01:
    The tester shall verify that the vendor has provided a description of the finite state model. This description shall contain the identification and description of all states of the module and a description of all corresponding state transitions. The tester shall verify that the descriptions of the state transitions include the internal module conditions, data inputs, and control inputs that cause transitions from one state to another, data outputs and status outputs resulting from transitions from one state to another.
  - TE11.16.01:
    The tester shall use the list supplied by the vendor to verify that a source listing for each software or firmware component is contained in the module.

Let us look at an example TE that is assessed by reviewing the vendor documentation, and this TE's associated AS and VE.

AS05.02 states, "The documentation requirements specified in {ISO/IEC 19790:2012} A.2.5 shall be provided." Following that, **VE05.02.01** states, "The vendor shall provide documentation as specified in ISO/IEC 19790:2012, A.2.5." Lastly, the **TE05.02.01** for this section states, "The tester shall verify completeness of the documentation specified in ISO/IEC 19790:2012, A.2.5." [6]

To fulfill **TE05.02.01**, the tester needs to check the documentation provided by the vendor and verify that it is present and complete. The example illustrates a documentation-type TE (i.e., 140A-TE). TEs of this type are ripe for automation because they only rely on checking for the presence of appropriate texts. The accuracy of the information provided for these TEs is later verified by subsequent tests and documentation reviews done during Functional Testing, Source Code Review, and Module Inspection.

By exploring the relationship between VEs and TEs, it becomes apparent that if some VEs were in the form of a standardized SP, then their corresponding TEs can be verified through automation. The NIST

CMVP worked hard on updating SP 800-140B and published its Revision 1 in November 2023. It specifies the expected content of the SP and provides an SP template for all vendors and labs to use.

The current NIST WebCryptik Br1 v1.0.3 has built-in MIS Tables and search capability to look up and select CAVP certificates. The completed MIS (Module Information Structure) Tables can be saved as a JSON file and combined with other information in an SP Microsoft Word template to build the final SP.

This TE WS explored an alternative method to generate the SP purely via JSON rather than implementing a hybrid approach that requires an SP Microsoft Word template to build the final SP. Following the CMVP's current SP Template v5.8, the NCCoE TE WS has developed an SP-evidence JSON file to satisfy all SP-TEs. The Protocol Workstream implemented the functionality on the ACMVP server, generating an SP based on the input SP-evidence JSON file. This functionality was demonstrated at the ICMC24.

Under the assumption that the SP strictly follows SP800-140Br1 [6] and the required SP content is captured in MIS Tables or the other data entries in the SP-evidence JSON file, all SP-TEs can reference the relevant data points in the SP-evidence JSON file. The existence of the reference can be automatically checked. If the reference exists, the corresponding TE passes.

SP-TEs must be satisfied by the information provided by the SP as specified in NIST SP800-140Br1, which we denote as **140B-TEs**. 140B-TEs is a subset of SP-TEs because a vendor may choose to include more information in the SP as required by the SP800-140Br1.

Furthermore, to maximize the automation, the 140A-TEs were classified under the mentioned types, where each type is captured by a standardized documentation-evidence JSON.

The following table lists all the TEs depending on the SP, regardless of whether the TE explicitly indicates the source of the vendor document to be SP or if SP800-140Br1 requires it, in column **SP-TEs**. The **non-140B-but-140A-TEs** column is intended not to duplicate the TEs from the SP-TEs column but to capture all other TEs that depend on vendor documentation, which could be SP, source code, FSM, Component List, design document, or other vendor proprietary documentation. Nevertheless, there are a few cases where the information needs to be in the SP and verified by (code) inspection or design document; the TEs (e.g., TE02.07.02) are listed under both columns, despite the duplication.

Table 1 Dividing 140A-TEs into non-140B-TEs and SP-TEs

FIPS 140-3 Section Title	140A-TEs	
	non-140B-but-140A-TEs	SP-TEs
General	None	None
Cryptographic Module Specification	TE02.03.02, TE02.07.01, TE02.07.02 (also SP-TE), TE02.10.01 (also SP-TE), TE02.10.02, TE02.13.02, TE02.17.09,	TE02.03.01, TE02.07.02, TE02.09.01, TE02.10.01, TE02.11.01, TE02.11.02, TE02.12.01, TE02.13.01, TE02.14.01, TE02.15.01, TE02.15.02, TE02.15.04, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09,

FIPS 140-3 Section Title	140A-TEs	
	non-140B-but-140A-TEs	SP-TEs
		TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.16.01, TE02.16.02, TE02.16.03, TE02.16.05, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.10, TE02.18.01, TE02.19.01, TE02.20.01, TE02.20.02, TE02.20.03, TE02.20.04, TE02.21.01, TE02.21.02, TE02.22.01, TE02.24.01, TE02.26.01, TE02.26.02, TE02.30.01
Cryptographic Module Interfaces	TE03.01.02 (also SP-TE), TE03.02.01, TE03.05.02, TE03.06.02, TE03.07.01, TE03.07.03, TE03.07.05, TE03.07.06, TE03.07.07, TE03.08.02, TE03.09.01, TE03.10.01, TE03.10.03, TE03.10.05, TE03.11.02, TE03.13.01, TE03.14.01, TE03.14.02, TE03.14.03, TE03.15.01, TE03.15.02, TE03.15.05, TE03.16.01, TE03.18.01, TE03.19.01, TE03.19.03,	TE03.01.01, TE03.01.02, TE03.01.03, TE03.02.02, TE03.03,01, TE03.04.01,
Roles, Services, and Authentication	TE04.02.01, TE04.03.01, TE04.07.01, TE04.07.02, TE04.19.01, TE04.20.01, TE04.20.02, TE04.21.01, TE04.22.01, TE04.25.01, TE04.33.01, TE04.35.01, TE04.38.01, TE04.39.01, TE04.42.01, TE04.42.02, TE04.43.01, TE04.44.01, TE04.45.01, TE04.51.02, TE04.53.01, TE04.54.01, TE04.55.01,	TE04.05.01, TE04.06.01, TE04.11.01, TE04.13.02, TE04.14.01, TE04.18.01, TE04.37.01, TE04.47.01, TE04.48.01, TE04.50.01, TE04.50.02, TE04.51.01, TE04.56.01, TE04.56.02, TE04.59.01
Software/Firmware Security	TE05.02.01, TE05.04.01, TE05.05.01, TE05.05.03, TE05.05.04, TE05.05.06, TE05.06.01, TE05.06.05, TE05.07.01, TE05.08.02,	TE05.05.02, TE05.17.01

FIPS 140-3 Section Title	140A-TEs	
	non-140B-but-140A-TEs	SP-TEs
	TE05.11.01, TE05.12.01, TE05.12.02, TE05.13.01, TE05.13.02, TE05.13.04, TE05.13.06, TE05.13.07, TE05.15.01, TE05.15.02, TE05.16.01, TE05.16.02, TE05.20.01, TE05.23.01	
Operational Environment	TE06.03.01, TE06.05.01, TE06.05.02, TE06.06.01, TE06.08.01, TE06.08.02, TE06.10.01, TE06.11.01, TE06.12.01, TE06.13.01, TE06.14.01, TE06.15.01, TE06.17.01, TE06.18.01, TE06.19.01, TE06.24.01, TE06.25.01, TE06.26.01, TE06.27.01, TE06.28.01,	TE06.07.01, TE06.09.01, TE06.20.01,
Physical Security	TE07.10.01, TE07.11.01, TE07.12.01, TE07.15.01, TE07.15.02, TE07.19.01, TE07.20.01, TE07.25.01, TE07.26.01, TE07.32.01, TE07.33.01, TE07.35.01, TE07.37.01, TE07.37.02, TE07.39.01, TE07.39.02, TE07.39.03, TE07.39.04, TE07.41.01, TE07.42.01, TE07.43.01, TE07.44.01, TE07.45.01, TE07.46.01, TE07.47.01, TE07.48.01, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.07, TE07.53.01, TE07.55.01, TE07.57.01, TE07.60.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.67.01, TE07.71.01, TE07.73.01,	TE07.01.01, TE07.09.01, TE07.09.02, TE07.19.01, TE07.26.02, TE07.77.04, TE07.81.03
Non-invasive Security	Not yet enforced by the CMVP	Not yet enforced by the CMVP
Sensitive Security Parameter Management	TE09.01.01, TE09.02.01, TE09.03.01, TE09.05.01,	TE09.04.01, TE09.04.02, TE09.06.01, TE09.06.02,

FIPS 140-3 Section Title	140A-TEs	
	non-140B-but-140A-TEs	**SP-TEs**
	TE09.08.02, TE09.14.01, TE09.16.01, TE09.16.02, TE09.21.01, TE09.23.01, TE09.23.02, TE09.23.04, TE09.24.01, TE09.25.01, TE09.27.01, TE09.28.06, TE09.29.01, TE09.29.02, TE09.31.01, TE09.32.01, TE09.36.01	TE09.06.03, TE09.07.01, TE09.08.01, TE09.09.01, TE09.09.02, TE09.10.01, TE09.10.02, TE09.13.01, TE09.13.02, TE09.19.01, TE09.22.01, TE09.28.01, TE09.28.05, TE09.33.01, TE09.37.01
Self-tests	TE10.12.01, TE10.12.02, TE10.15.01, TE10.15.02, TE10.20.01, TE10.21.01, TE10.21.02, TE10.22.02, TE10.22.03, TE10.22.05, TE10.27.01, TE10.28.01, TE10.29.01, TE10.33.02, TE10.34.02, TE10.35.01, TE10.35.02, TE10.35.03, TE10.37.03, TE10.37.04, TE10.37.07, TE10.37.08, TE10.46.01, TE10.46.02, TE10.48.02, TE10.49.02, TE10.51.01, TE10.51.02, TE10.51.03	TE10.07.01, TE10.07.02, TE10.08.01, TE10.08.02, TE10.09.01, TE10.09.02, TE10.24.01, TE10.25.01, TE10.33.01, TE10.34.01, TE10.37.01, TE10.37.02, TE10.53.01
Life-cycle Assurance	TE11.01.01, TE11.03.01, TE11.04.01, TE11.04.02, TE11.04.03, TE11.04.04, TE11.05.01, TE11.06.01, TE11.08.01, TE11.08.02, TE11.08.03, TE11.08.04, TE11.08.05, TE11.08.07, TE11.08.08, TE11.08.10, TE11.08.11, TE11.08.12, TE11.13.01, TE11.15.01, TE11.15.02, TE11.16.01, TE11.17.01, TE11.18.01, TE11.19.01, TE11.21.01, TE11.23.01, TE11.24.01, TE11.25.01, TE11.26.01, TE11.28.01, TE11.28.02, TE11.28.03, TE11.29.01, TE11.29.02, TE11.30.01, TE11.31.01, TE11.33.01, TE11.34.01, TE11.38.03,	TE11.32.01, TE11.35.01, TE11.36.01, TE11.37.01, TE11.38.01, TE11.39.01
Mitigation of Other Attacks	TE12.01.01, TE12.04.02,	TE12.02.01, TE12.04.01, TE12.04.03

FIPS 140-3 Section Title	140A-TEs	
	non-140B-but-140A-TEs	SP-TEs
NIST SP 800-140A	TEA01.01	
NIST SP 800-140B (Cryptographic module security policy)		TEB.01.01, TEB.02.01, TEB.03.01, TEB.03.02

3.2.1.1 TEs depending on Security Policy

See the SP-TEs column in Table 1. Dividing 140A-TEs into non-140B-TEs and SP-TEs.

3.2.1.2 TEs requiring source code inspection

TEs depending on source code review or inspection are a subset of the non-140B-but-140A-TEs column in Table 1 – Dividing 140A-TEs into non-140B-TEs and SP-TEs. Some TEs have the explicit wording of "code" or "source code", while others imply it via the phrase, "by inspection" or "inspecting the module." TEs requiring source code review are tagged as SC-TE in Appendix B. CMVP TEs Tables of this document.

3.2.1.3 TEs depending on other documents

TEs requiring other documents are tagged as OD-TE in section 3.2.3.

3.2.2 TEs Requiring Module Functional Test

TEs in this category require the tester to exercise and manipulate the module to test its functionality. To do this, testers rely on various pieces of evidence that include logfile names, screenshots, or remote testing/video observation. In essence, the tester must directly see and interact with the module to ensure that it functions in the way specified by the vendor.

TE09.03.02 is an example of this category. It states: "For each [Sensitive Security Parameter (SSP)] that can be entered, the tester shall first enter the SSP while assuming the correct entity. The tester shall then verify that entry is not possible when assuming an incorrect entity"[3]. To fulfill this TE, the tester must assume specific entities and use the module as those assumed roles, testing that the module correctly identifies roles and grants only the appropriate SSP entry service to each entity.

This category of TEs is the hardest to automate; however, we may address the work surrounding functional testing. Automation opportunities may be found in how the lab collects and prepares the test evidence (e.g., log files) from functional testing.

Table 2 below lists all TEs that require Functional Testing.

Table 2 TEs Requiring Functional Testing

FIPS 140-3 Section Name	TEs for SL 1-4	TEs for SL 2-4	TEs for SL 3-4	TEs for SL 4
General	N/A			

FIPS 140-3 Section Name	TEs for SL 1-4	TEs for SL 2-4	TEs for SL 3-4	TEs for SL 4
Module Specification	TE02.10.01 (or SC-TE), TE02.12.01, TE02.13.03, TE02.15.03, TE02.15.05, TE02.16.04, TE02.17.02, TE02.17.04, TE02.19.02, TE02.22.02, TE02.24.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.02	None	None	None
Module Interfaces	TE03.01.04, TE03.02.01, TE03.05.01, TE03.05.02, TE03.06.01, TE03.06.02, TE03.07.02, TE03.07.04, TE03.07.08, TE03.08.01, TE03.08.02, TE03.09.02, TE03.10.02, TE03.10.04, TE03.11.01, TE03.11.03, TE03.13.02, TE03.14.03, TE03.15.02, TE03.15.03, TE03.15.04, TE03.15.06,	None	TE03.16.01 (or SC-TE), TE03.18.01, TE03.18.02, TE03.19.02, TE03.19.04, TE03.20.01, TE03.21.01,	TE03.22.01
Roles, Services, and Authentication	TE04.02.02, TE04.02.03, TE04.07.03, TE04.11.02, TE04.13.01, TE04.13.03, TE04.14.02, TE04.15.01, TE04.18.01, TE04.19.02, TE04.19.03, TE04.20.01, TE04.20.03, TE04.21.02, TE04.22.02, TE04.23.01, TE04.25.02, TE04.25.03, TE04.28.01, TE04.29.01, TE04.32.01, TE04.33.01, TE04.34.01, TE04.35.02, TE04.43.02, TE04.44.02, TE04.56.02 (L1 only)	TE04.37.02, TE04.38.02, TE04.45.02, TE04.45.03, TE04.52.01, TE04.53.01 (L2 only) TE04.54.02, TE04.54.03, TE04.55.02	TE04.39.02, TE04.39.03, TE04.39.04, TE04.42.03, TE04.42.04,	TE04.59.01
Software/Firmware Security	TE05.05.05, TE05.05.07, TE05.06.02, TE05.06.03, TE05.06.04, TE05.06.06, TE05.07.01, TE05.08.01, TE05.08.02, TE05.11.01, TE05.11.02, TE05.12.02, TE05.13.01, TE05.13.02, TE05.13.03, TE05.13.04, TE05.13.05, TE05.13.06, TE05.13.08,	TE05.15.01, TE05.15.02, TE05.16.03, TE05.17.02	TE05.20.01, TE05.23.01	none
Operational Environment	TE06.05.01, TE06.05.02, TE06.05.03, TE06.06.01,	The following TEs are for L2 only:	None	None

FIPS 140-3 Section Name	TEs for SL 1-4	TEs for SL 2-4	TEs for SL 3-4	TEs for SL 4
	TE06.06.02, TE06.08.01, TE06.08.02, TE06.08.03,	TE06.09.02, TE06.09.03, TE06.10.01, TE06.10.02, TE06.10.03, TE06.11.01, TE06.11.02, TE06.11.03, TE06.12.01, TE06.12.02, TE06.12.03, TE06.13.01, TE06.13.02, TE06.13.03, TE06.14.01, TE06.14.02, TE06.14.03, TE06.15.01, TE06.15.02, TE06.15.03, TE06.17.01, TE06.17.02, TE06.17.03, TE06.18.01, TE06.18.02, TE06.18.03, TE06.24.01, TE06.25.01, TE06.25.02, TE06.26.01, TE06.26.02, TE06.27.01, TE06.27.02, TE06.28.01, TE06.28.02, TE06.28.03, TE06.28.04		
Physical Security	TE07.01.02, TE07.10.02, TE07.11.02, TE07.13.01, TE07.15.01, TE07.37.01, TE07.43.01, TE07.60.01,	TE07.19.01, TE07.20.01, TE07.35.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.62.01,	TE07.25.01, TE07.26.01, TE07.27.01, TE07.37.03, TE07.39.03, TE07.39.04, TE07.39.05, TE07.39.06, TE07.50.02, TE07.50.03, TE07.51.04, TE07.51.05,	TE07.32.01, TE07.41.01, TE07.41.02, TE07.42.02, TE07.53.01, TE07.55.01, TE07.58.01, TE07.67.01, TE07.71.02,

FIPS 140-3 Section Name	TEs for SL 1-4	TEs for SL 2-4	TEs for SL 3-4	TEs for SL 4
		TE07.63.01,	TE07.51.06, TE07.51.08, TE07.51.09, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.08, TE07.65.09, TE07.77.01, TE07.77.02, TE07.77.03, TE07.81.01, TE07.81.02	
Non-Invasive Security	N/A			
SSP Management	TE09.01.02, TE09.01.03, TE09.02.02, TE09.03.02, TE09.03.03, TE09.13.03, TE09.14.02, TE09.16.03, TE09.18.01, TE09.18.02, TE09.21.02, TE09.21.03, TE09.21.04, TE09.22.01, TE09.24.02, TE09.25.02, TE09.27.02, TE09.28.02, TE09.28.03, TE09.28.04, TE09.33.02, TE09.36.02, TE09.37.02	None	None	None
Self-Tests	TE10.07.03, TE10.07.04, TE10.07.05, TE10.08.03, TE10.09.03, TE10.10.01, TE10.10.02, TE10.11.01, TE10.15.01, TE10.15.02, TE10.21.01, TE10.21.02, TE10.21.03, TE10.21.04, TE10.22.01, TE10.22.04, TE10.25.02, TE10.27.01, TE10.28.02, TE10.34.03, TE10.35.04, TE10.37.05, TE10.37.06, TE10.37.09, TE10.46.03, TE10.46.04, TE10.48.01, TE10.48.03, TE10.49.01, TE10.49.03, TE10.53.02, TE10.53.03		TE10.12.03, TE10.12.04, TE10.12.05, TE10.54.01	
Life-Cycle Assurance	TE11.08.06, TE11.08.09, TE11.11.01, TE11.13.02, TE11.32.02			TE11.28.02, TE11.28.03, TE11.28.04

FIPS 140-3 Section Name	TEs for SL 1-4	TEs for SL 2-4	TEs for SL 3-4	TEs for SL 4
Mitigation of Other Attacks	N/A			

3.2.3 A Complete list of TE Classification

All TEs are classified into four categories (i.e., SP-TE, OD-TE, SC-TC, FT-TE) and their potential combination.

The complete set of the TE classification tags is listed below:

- SP-TE: TEs depending on the SP
- SC-TE: TEs depending on source code review or inspection
- OD-TE: TEs depending on other vendor documentation
- FT-TE: TEs depending on functional testing evidence
- SC-TE/OD-TE: TEs depend on vendor documentation, regardless of whether it is source code or not
- FT-TE/OD-TE: TEs depend on vendor documentation, regardless of whether it is functional testing evidence or not
- SC-TE/SP-TE: TEs depending on source code review or on the SP
- SP-TE, FT-TE: TE depending on the SP and on functional testing
- SC-TE, FT-TE: TE depending on source code review and on functional testing

Greyed out TEs are those not currently required by the CMVP.

The OD-TEs depend on proprietary vendor documentation. Therefore, they do not belong to the SP-TE category.

Examples:

- FT-TE:
  - The tester shall verify, by exercising the module, that the status indicator is provided when the trusted channel is in use. (e.g., TE03.21.01)
  - The tester shall verify that an identity-based authentication mechanism is employed for all services utilizing the trusted channel. (e.g., TE03.20.01)
- SP-TE, FT-TE or FT-TE, OD-TE:
  - “The tester shall use the vendor documentation to access multi-factor identity-based authentication.” (e.g., TE04.59.01)
  - The tester shall verify from the vendor documentation and by inspection that the approved authentication mechanism implemented in the operating system meets the applicable requirements. (TE04.53.01)
- SP-TE, FT-TE or FT-TE, OD-TE:
  - “The tester shall invoke the approved mode of operation using the vendor-provided instructions found in the non-proprietary security policy” (e.g., TE02.19.02)
  - “The tester shall verify that the module implements a bypass capability as specified in the vendor documentation” (e.g., TE04.18.01)

- "The tester shall attempt to perform cryptographic operations using each of the SSPs that were stored in the module." (e.g., TE04.35.02)
- "The tester shall perform the following tests." (e.g., TE04.38.02)
- "The tester shall exercise the cryptographic module" (e.g., TE04.42.04)

Legend of TE Tags

Table 3 Legend of TE Tags

SP-TE: SP-dependent TE	FT-TE/OD-TE: Functional-Test- or documentation-dependent TE
SC-TE: Source-Code-dependent TE	SC-TE/SP-TE: Source-Code- or SP-dependent TE
OD-TE: Other-Document-dependent TE	SP-TE, FT-TE: SP- and Functional Test-dependent TE
FT-TE: Functional-Test-dependent TE	SC-TE, FT-TE: Source-Code- and Functional-Test-dependent TE
SC-TE/OD-TE: Source-Code- or documentation-dependent TE	

TE02.03.01	SP-TE
TE02.03.02	OD-TE
TE02.07.01	SC-TE, SP-TE
TE02.07.02	SC-TE, SP-TE
TE02.09.01	SP-TE
TE02.10.01	SP-TE, SC-TE/FT-TE
TE02.10.02	OD-TE
TE02.11.01	SP-TE
TE02.11.02	SP-TE
TE02.12.01	SP-TE, FT-TE
TE02.13.01	SP-TE
TE02.13.02	OD-TE
TE02.13.03	FT-TE
TE02.14.01	SP-TE
TE02.15.01	SP-TE
TE02.15.02	SP-TE
TE02.15.03	FT-TE
TE02.15.04	SP-TE
TE02.15.05	FT-TE
TE02.15.06	SP-TE
TE02.15.07	SP-TE
TE02.15.08	SP-TE
TE02.15.09	SP-TE
TE02.15.10	SP-TE
TE02.15.11	SP-TE
TE02.15.12	SP-TE
TE02.15.13	SP-TE
TE02.15.14	SP-TE
TE02.16.01	SP-TE
TE02.16.02	SP-TE
TE02.16.03	SP-TE
TE02.16.04	FT-TE
TE02.16.05	SP-TE
TE02.17.01	SP-TE
TE02.17.02	SP-TE, FT-TE
TE02.17.03	SP-TE
TE02.17.04	FT-TE
TE02.17.05	SP-TE
TE02.17.06	SP-TE
TE02.17.07	SP-TE
TE02.17.08	SP-TE
TE02.17.09	OD-TE
TE02.17.10	SP-TE
TE02.18.01	SP-TE
TE02.19.01	SP-TE
TE02.19.02	FT-TE
TE02.20.01	SP-TE
TE02.20.02	SP-TE
TE02.20.03	SP-TE
TE02.20.04	SP-TE
TE02.21.01	SP-TE
TE02.21.02	SP-TE
TE02.22.01	SP-TE
TE02.22.02	FT-TE
TE02.24.01	SP-TE
TE02.24.02	FT-TE
TE02.26.01	SP-TE
TE02.26.02	SP-TE
TE02.26.03	FT-TE
TE02.26.04	FT-TE
TE02.26.05	FT-TE
TE02.28.01	FT-TE
TE02.28.02	FT-TE
TE02.30.01	SP-TE
TE02.30.02	FT-TE
TE03.01.01	SP-TE
TE03.01.02	SP-TE, SC-TE
TE03.01.03	SP-TE
TE03.01.04	FT-TE
TE03.02.01	SC-TE, FT-TE
TE03.02.02	SP-TE
TE03.03.01	SP-TE
TE03.04.01	SP-TE
TE03.05.01	FT-TE
TE03.05.02	OD-TE, FT-TE
TE03.06.01	FT-TE
TE03.06.02	OD-TE, FT-TE
TE03.07.01	OD-TE
TE03.07.02	FT-TE
TE03.07.03	OD-TE
TE03.07.04	FT-TE
TE03.07.05	OD-TE, SC-TE
TE03.07.06	OD-TE
TE03.07.07	OD-TE
TE03.07.08	FT-TE
TE03.08.01	FT-TE
TE03.08.02	FT-TE, OD-TE
TE03.09.01	OD-TE
TE03.09.02	FT-TE
TE03.10.01	OD-TE
TE03.10.02	FT-TE
TE03.10.03	OD-TE
TE03.10.04	FT-TE
TE03.10.05	SC-TE/OD-TE
TE03.11.01	FT-TE
TE03.11.02	OD-TE
TE03.11.03	FT-TE
TE03.13.01	OD-TE
TE03.13.02	FT-TE
TE03.14.01	SC-TE/OD-TE
TE03.14.02	SC-TE/OD-TE
TE03.14.03	FT-TE, SC-TE
TE03.15.01	OD-TE
TE03.15.02	FT-TE, SC-TE
TE03.15.03	FT-TE
TE03.15.04	FT-TE
TE03.15.05	SC-TE
TE03.15.06	FT-TE
TE03.16.01	OD-TE, SC-TE/FT-TE
TE03.18.01	OD-TE, FT-TE
TE03.18.02	FT-TE
TE03.19.01	OD-TE, SC-TE
TE03.19.02	FT-TE
TE03.19.03	OD-TE
TE03.19.04	FT-TE

TE03.20.01	FT-TE
TE03.21.01	FT-TE
TE03.22.01	FT-TE
TE04.02.01	OD-TE
TE04.02.02	FT-TE
TE04.02.03	FT-TE
TE04.03.01	OD-TE
TE04.05.01	SP-TE
TE04.06.01	SP-TE
TE04.07.01	OD-TE
TE04.07.02	OD-TE
TE04.07.03	FT-TE
TE04.11.01	SP-TE
TE04.11.02	FT-TE
TE04.13.01	FT-TE
TE04.13.02	SP-TE
TE04.13.03	FT-TE, ~~OD-TE~~
TE04.14.01	SP-TE
TE04.14.02	FT-TE
TE04.15.01	FT-TE
TE04.18.01	FT-TE, OD-TE
TE04.19.01	OD-TE
TE04.19.02	FT-TE
TE04.19.03	FT-TE
TE04.20.01	FT-TE, OD-TE
TE04.20.02	OD-TE
TE04.20.03	FT-TE
TE04.21.01	OD-TE
TE04.21.02	FT-TE
TE04.22.01	OD-TE
TE04.22.02	FT-TE
TE04.23.01	FT-TE
TE04.25.01	OD-TE
TE04.25.02	FT-TE
TE04.25.03	FT-TE
TE04.28.01	FT-TE
TE04.29.01	FT-TE
TE04.32.01	FT-TE
TE04.33.01	FT-TE, OD-TE
TE04.34.01	FT-TE
TE04.35.01	OD-TE
TE04.35.02	FT-TE
TE04.37.01	SP-TE
TE04.37.02	FT-TE
TE04.38.01	OD-TE
TE04.38.02	FT-TE
TE04.39.01	OD-TE
TE04.39.02	FT-TE
TE04.39.03	FT-TE
TE04.39.04	FT-TE
TE04.42.01	OD-TE
TE04.42.02	OD-TE
TE04.42.03	FT-TE
TE04.42.04	FT-TE
TE04.43.01	OD-TE
TE04.43.02	FT-TE
TE04.44.01	OD-TE
TE04.44.02	FT-TE
TE04.45.01	OD-TE
TE04.45.02	FT-TE
TE04.45.03	FT-TE
TE04.47.01	SP-TE
TE04.48.01	SP-TE
TE04.50.01	SP-TE
TE04.50.02	SP-TE
TE04.51.01	SP-TE
TE04.51.02	SP-TE
TE04.52.01	OD-TE, FT-TE
TE04.53.01	OD-TE, FT-TE
TE04.54.01	OD-TE
TE04.54.02	FT-TE
TE04.54.03	FT-TE
TE04.55.01	OD-TE
TE04.55.02	FT-TE
TE04.56.01	SP-TE
TE04.56.02	FT-TE
TE04.59.01	SP-TE, FT-TE
TE05.02.01	OD-TE
TE05.04.01	SC-TE
TE05.05.01	SC-TE
TE05.05.02	SP-TE
TE05.05.03	OD-TE
TE05.05.04	OD-TE
TE05.05.05	FT-TE
TE05.05.06	SC-TE/OD-TE
TE05.05.07	FT-TE
TE05.06.01	SC-TE
TE05.06.02	FT-TE
TE05.06.03	FT-TE
TE05.06.04	FT-TE
TE05.06.05	SC-TE
TE05.06.06	FT-TE
TE05.07.01	OD-TE, FT-TE
TE05.08.01	FT-TE
TE05.08.02	FT-TE, SC-TE
TE05.11.01	FT-TE
TE05.11.02	FT-TE
TE05.12.01	OD-TE
TE05.12.02	FT-TE, OD-TE
TE05.13.01	FT-TE, OD-TE
TE05.13.02	FT-TE, OD-TE
TE05.13.03	FT-TE
TE05.13.04	FT-TE, OD-TE
TE05.13.05	FT-TE
TE05.13.06	FT-TE, OD-TE
TE05.13.07	SC-TE/OD-TE
TE05.13.08	FT-TE
TE05.15.01	FT-TE, OD-TE
TE05.15.02	FT-TE, OD-TE
TE05.16.01	OD-TE
TE05.16.02	OD-TE
TE05.16.03	FT-TE
TE05.17.01	SP-TE
TE05.17.02	FT-TE
TE05.20.01	SC-TE, FT-TE
TE05.23.01	FT-TE/OD-TE

TE06.03.01	OD-TE
TE06.05.01	OD-TE, FT-TE
TE06.05.02	OD-TE, FT-TE
TE06.05.03	FT-TE
TE06.06.01	OD-TE, FT-TE
TE06.06.02	FT-TE
TE06.07.01	SP-TE
TE06.08.01	OD-TE, FT-TE
TE06.08.02	OD-TE, FT-TE
TE06.08.03	FT-TE
TE06.09.01	SP-TE
TE06.09.02	FT-TE
TE06.09.03	FT-TE
TE06.10.01	OD-TE, FT-TE
TE06.10.02	FT-TE
TE06.10.03	FT-TE
TE06.11.01	OD-TE, FT-TE
TE06.11.02	FT-TE
TE06.11.03	FT-TE
TE06.12.01	OD-TE, FT-TE
TE06.12.02	FT-TE
TE06.12.03	FT-TE
TE06.13.01	OD-TE, FT-TE
TE06.13.02	FT-TE
TE06.13.03	FT-TE
TE06.14.01	OD-TE, FT-TE
TE06.14.02	FT-TE
TE06.14.03	FT-TE
TE06.15.01	OD-TE, FT-TE
TE06.15.02	FT-TE
TE06.15.03	FT-TE
TE06.17.01	OD-TE, FT-TE
TE06.17.02	FT-TE
TE06.17.03	FT-TE
TE06.18.01	OD-TE, FT-TE
TE06.18.02	FT-TE
TE06.18.03	FT-TE
TE06.19.01	OD-TE
TE06.20.01	SP-TE
TE06.24.01	OD-TE, FT-TE
TE06.25.01	OD-TE, FT-TE
TE06.25.02	FT-TE
TE06.26.01	OD-TE, FT-TE
TE06.26.02	FT-TE
TE06.27.01	OD-TE, FT-TE
TE06.27.02	FT-TE
TE06.28.01	OD-TE, FT-TE
TE06.28.02	FT-TE
TE06.28.03	FT-TE
TE06.28.04	FT-TE
TE07.01.01	SP-TE
TE07.01.02	FT-TE
TE07.09.01	SP-TE
TE07.09.02	SP-TE
TE07.10.01	OD-TE
TE07.10.02	FT-TE
TE07.11.01	OD-TE
TE07.11.02	FT-TE
TE07.12.01	OD-TE
TE07.13.01	FT-TE
TE07.15.01	FT-TE, OD-TE
TE07.15.02	OD-TE
TE07.19.01	FT-TE, OD-TE
TE07.20.01	FT-TE, OD-TE
TE07.25.01	FT-TE, OD-TE
TE07.26.01	OD-TE FT-TE
TE07.26.02	SP-TE
TE07.27.01	FT-TE
TE07.32.01	OD-TE, FT-TE
TE07.33.01	OD-TE
TE07.35.01	FT-TE, OD-TE
TE07.37.01	FT-TE, OD-TE
TE07.37.02	OD-TE
TE07.37.03	FT-TE
TE07.39.01	OD-TE
TE07.39.02	OD-TE
TE07.39.03	FT-TE, OD-TE
TE07.39.04	FT-TE, OD-TE
TE07.39.05	FT-TE
TE07.39.06	FT-TE
TE07.41.01	FT-TE, OD-TE
TE07.41.02	FT-TE
TE07.42.01	OD-TE
TE07.42.02	FT-TE
TE07.43.01	FT-TE, OD-TE
TE07.44.01	FT-TE, OD-TE
TE07.45.01	FT-TE, OD-TE
TE07.45.02	FT-TE
TE07.46.01	FT-TE, OD-TE
TE07.47.01	FT-TE, OD-TE
TE07.47.02	FT-TE
TE07.48.01	FT-TE, OD-TE
TE07.48.02	FT-TE
TE07.50.01	OD-TE
TE07.50.02	FT-TE, OD-TE
TE07.50.03	FT-TE, OD-TE
TE07.51.01	OD-TE
TE07.51.02	OD-TE
TE07.51.03	OD-TE
TE07.51.04	FT-TE, OD-TE
TE07.51.05	FT-TE, OD-TE
TE07.51.06	FT-TE
TE07.51.07	OD-TE
TE07.51.08	FT-TE
TE07.51.09	FT-TE
TE07.53.01	OD-TE, FT-TE
TE07.55.01	OD-TE, FT-TE
TE07.57.01	OD-TE
TE07.58.01	FT-TE
TE07.60.01	FT-TE, OD-TE
TE07.62.01	FT-TE
TE07.63.01	FT-TE
TE07.65.01	OD-TE
TE07.65.02	OD-TE
TE07.65.03	OD-TE
TE07.65.04	FT-TE, OD-TE
TE07.65.05	FT-TE, OD-TE

TE07.65.06	FT-TE, OD-TE
TE07.65.07	OD-TE
TE07.65.08	FT-TE
TE07.65.09	FT-TE
TE07.67.01	OD-TE, FT-TE
TE07.71.01	OD-TE
TE07.71.02	FT-TE
TE07.73.01	OD-TE
TE07.77.01	FT-TE
TE07.77.02	FT-TE
TE07.77.03	FT-TE
TE07.77.04	SP-TE
TE07.81.01	FT-TE
TE07.81.02	FT-TE
TE07.81.03	SP-TE
TE08.03.01	OD-TE
TE08.04.01	OD-TE
TE08.05.01	OD-TE
TE08.06.01	OD-TE
TE08.07.01	OD-TE
TE09.01.01	OD-TE
TE09.01.02	FT-TE
TE09.01.03	FT-TE
TE09.02.01	OD-TE
TE09.02.02	FT-TE
TE09.03.01	OD-TE
TE09.03.02	FT-TE
TE09.03.03	FT-TE
TE09.04.01	SP-TE
TE09.04.02	SP-TE
TE09.05.01	OD-TE
TE09.06.01	SP-TE
TE09.06.02	SP-TE
TE09.06.03	SP-TE
TE09.07.01	SP-TE
TE09.08.01	SP-TE
TE09.08.02	OD-TE
TE09.09.01	SP-TE
TE09.09.02	SP-TE
TE09.10.01	SP-TE
TE09.10.02	SP-TE
TE09.13.01	SP-TE
TE09.13.02	SP-TE
TE09.13.03	FT-TE
TE09.14.01	OD-TE
TE09.14.02	FT-TE
TE09.16.01	OD-TE
TE09.16.02	OD-TE
TE09.16.03	FT-TE
TE09.18.01	FT-TE
TE09.18.02	FT-TE
TE09.19.01	SP-TE
TE09.21.01	OD-TE
TE09.21.02	FT-TE
TE09.21.03	FT-TE
TE09.21.04	FT-TE
TE09.22.01	FT-TE
TE09.23.01	OD-TE
TE09.23.02	OD-TE
TE09.23.03	FT-TE/OD-TE
TE09.23.04	OD-TE
TE09.24.01	OD-TE
TE09.24.02	FT-TE
TE09.25.01	OD-TE
TE09.25.02	FT-TE
TE09.27.01	OD-TE
TE09.27.02	FT-TE
TE09.28.01	SP-TE
TE09.28.02	FT-TE
TE09.28.03	FT-TE
TE09.28.04	FT-TE
TE09.28.05	SP-TE
TE09.28.06	OD-TE
TE09.29.01	OD-TE
TE09.29.02	OD-TE
TE09.31.01	OD-TE
TE09.32.01	OD-TE
TE09.33.01	SP-TE
TE09.33.02	FT-TE
TE09.36.01	OD-TE
TE09.36.02	FT-TE
TE09.37.01	SP-TE
TE09.37.02	FT-TE
TE10.07.01	SP-TE
TE10.07.02	SP-TE
TE10.07.03	FT-TE
TE10.07.04	FT-TE
TE10.07.05	FT-TE/SC-TE
TE10.08.01	SP-TE
TE10.08.02	SP-TE
TE10.08.03	FT-TE
TE10.09.01	SP-TE
TE10.09.02	SP-TE
TE10.09.03	FT-TE
TE10.10.01	FT-TE
TE10.10.02	FT-TE
TE10.11.01	FT-TE
TE10.12.01	OD-TE
TE10.12.02	OD-TE
TE10.12.03	FT-TE
TE10.12.04	FT-TE
TE10.12.05	FT-TE
TE10.15.01	OD-TE, FT-TE
TE10.15.02	SC-TE/OD-TE, FT-TE
TE10.20.01	SC-TE/OD-TE
TE10.21.01	OD-TE, FT-TE
TE10.21.02	FT-TE, OD-TE
TE10.21.03	FT-TE
TE10.21.04	FT-TE
TE10.22.01	FT-TE
TE10.22.02	SC-TE/OD-TE
TE10.22.03	SC-TE/OD-TE
TE10.22.04	FT-TE
TE10.22.05	SC-TE/OD-TE
TE10.24.01	SP-TE
TE10.24.02	SC-TE/OD-TE
TE10.25.01	SP-TE
TE10.25.02	FT-TE

TE10.27.01	FT-TE, OD-TE	**TE11.03.01**	OD-TE	**TE11.30.01**	OD-TE
TE10.28.01	OD-TE, SC-TE	**TE11.04.01**	OD-TE	**TE11.31.01**	OD-TE
TE10.28.02	FT-TE	**TE11.04.02**	OD-TE	**TE11.32.01**	SP-TE
TE10.29.01	SC-TE, OD-TE	**TE11.04.03**	OD-TE	**TE11.32.02**	FT-TE
TE10.33.01	SP-TE	**TE11.04.04**	OD-TE	**TE11.33.01**	OD-TE
TE10.33.02	SC-TE/OD-TE	**TE11.05.01**	OD-TE	**TE11.34.01**	OD-TE
TE10.34.01	SP-TE	**TE11.06.01**	OD-TE	**TE11.35.01**	SP-TE
TE10.34.02	OD-TE, SC-TE	**TE11.08.01**	OD(FSM)-TE	**TE11.36.01**	SP-TE
TE10.34.03	FT-TE	**TE11.08.02**	OD(FSM)-TE	**TE11.37.01**	SP-TE
TE10.35.01	OD-TE, SC-TE	**TE11.08.03**	OD(FSM)-TE	**TE11.38.01**	SP-TE
TE10.35.02	OD-TE, SC-TE	**TE11.08.04**	OD(FSM)-TE	**TE11.38.03**	OD-TE
TE10.35.03	OD-TE, SC-TE	**TE11.08.05**	OD(FSM)-TE	**TE11.39.01**	SP-TE
TE10.35.04	FT-TE	**TE11.08.06**	FT-TE	**TE12.01.01**	OD-TE
TE10.37.01	SP-TE	**TE11.08.07**	OD(FSM)-TE	**TE12.02.01**	SP-TE
TE10.37.02	SP-TE	**TE11.08.08**	OD(FSM)-TE	**TE12.04.01**	SP-TE
TE10.37.03	OD-TE	**TE11.08.09**	FT-TE	**TE12.04.02**	OD-TE
TE10.37.04	SC-TE/OD-TE	**TE11.08.10**	OD(FSM)-TE	**TE12.04.03**	SP-TE
TE10.37.05	FT-TE	**TE11.08.11**	OD(FSM)-TE	**TEA01.01**	OD-TE
TE10.37.06	FT-TE	**TE11.08.12**	OD(FSM)-TE	**TEB01.01**	SP-TE
TE10.37.07	SC-TE/OD-TE	**TE11.11.01**	FT-TE	**TEB02.01**	SP-TE
TE10.37.08	SC-TE/OD-TE	**TE11.13.01**	OD(FSM)-TE	**TEB03.01**	SP-TE
TE10.37.09	FT-TE	**TE11.13.02**	FT-TE	**TEB03.02**	SP-TE
TE10.46.01	OD-TE	**TE11.15.01**	OD-TE		
TE10.46.02	SC-TE, OD-TE	**TE11.15.02**	OD-TE		
TE10.46.03	FT-TE	**TE11.16.01**	SC-TE/OD-TE		
TE10.46.04	FT-TE	**TE11.17.01**	SC-TE/OD-TE		
TE10.48.01	FT-TE	**TE11.18.01**	SC-TE/OD-TE		
TE10.48.02	SC-TE, OD-TE	**TE11.19.01**	OD-TE		
TE10.48.03	FT-TE	**TE11.21.01**	OD-TE		
TE10.49.01	FT-TE	**TE11.23.01**	OD-TE		
TE10.49.02	SC-TE, OD-TE	**TE11.24.01**	SC-TE		
TE10.49.03	FT-TE	**TE11.25.01**	OD-TE		
TE10.51.01	SC-TE, OD-TE	**TE11.26.01**	OD-TE		
TE10.51.02	SC-TE, OD-TE	**TE11.28.01**	SC-TE		
TE10.51.03	SC-TE, OD-TE	**TE11.28.02**	FT-TE/OD-TE, SC-TE		
TE10.53.01	SP-TE	**TE11.28.03**	FT-TE/OD-TE, SC-TE		
TE10.53.02	FT-TE	**TE11.28.04**	FT-TE/OD-TE		
TE10.53.03	FT-TE	**TE11.29.01**	OD-TE		
TE10.54.01	FT-TE, SC-TE	**TE11.29.02**	OD-TE		
TE11.01.01	OD-TE				

724

3.3 Test Methods for Functional Testing TEs

The diverse set of cryptographic modules and their varying restrictive operating environments can create challenges in choosing the right approach and selecting an appropriate toolset to capture the evaluation TE. The CMVP provides some limited guidance, but it is necessary to identify which test methods are relevant to the granularity of individual TEs.

3.3.1 Testing Access

There is frequently a challenge in accessing the operational environment for effective testing of a cryptographic module. There are allowances for various methodologies to follow for accommodating these challenges. For any given evaluation, it is assumed by default that the Testing Access used for all TEs is the same; however, any given TE might in fact require an alternate allowed Testing Access method to be used.

The Testing Access methods are as follows:

Physical: Testing a module directly by lab personnel within a controlled lab environment.

Remote: Testing a module remotely while obtaining the equivalent assurance as if the test were performed at the vendor's facility.

Observed: Testing a module by vendor personnel within a controlled lab environment while lab personnel observe the triggering and responses of the module.

3.3.2 Selection Criteria

The challenge is to assign only the appropriate test methods to each of the identified TEs. Drawing from CMVP, lab, and original vendor expertise, the criteria can be used to refine the test methods to be used for each TE.

Test methods are the defined techniques that can be utilized while ensuring confidence in capturing actual module operation under real-world conditions and enabling an efficient evidence-gathering workflow. Only a limited set of test method categories exists for the team to focus on in their pursuit, which can best be described as:

- **Debugger:** The ability to run or halt the target program using breakpoints, step through code line by line, and display or modify the contents of memory, CPU registers, and stack frames.
- **Simulation:** Imitates the representation of the functioning of one system or process by means of the functioning of another.
- **Emulation:** Hardware or software that permits programs written for one environment to be run unaltered on another environment.
- **Harness:** Hardware or software that manipulates an operating environment with the purpose of triggering events and capturing the corresponding responses or results.
- **Manual:** Action(s) by a user to perform a set of designated steps for the purpose of triggering events and capturing the corresponding responses or results.

- **Other:** Due to the diversity and complexity of operating environments, the toolset needed to perform the gathering of relevant TE may not fit precisely within the above five test methods. This warrants the need for a catch-all method that enables the tester to comprehensively describe the methodology used to capture the TE.

3.3.2.1 Debugger

No clearly articulated interpretation of when and how a debugger can and should be used is available, as much of what is known comes from lab empirical evidence.

3.3.2.2 Simulation and/or Emulation

Drawing from guidance currently provided by CMVP in the Management Manual, dated 10/16/2025, Version 2.5, labs may apply emulators or simulators, depending on the type of testing results to be achieved. The three broad areas of focus during the testing of a cryptographic module are operational testing of the module at the module's defined boundary, operational fault induction testing, and algorithm testing.

1. **Operational Testing** – Emulation or simulation is prohibited for the operational testing of a cryptographic module. Actual testing of the cryptographic module must be performed utilizing the defined ports, interfaces, and services that a module provides. A test harness or a modified version to induce an error may be utilized; however, no changes to code or circuitry responsible for the tested response may be made.
2. **Operational Fault Induction Testing** – An emulator or simulator may be utilized for fault induction to test a cryptographic module's transition to error states as a complement to the source code review. Rationale must be provided for the applicable TE as to why a method does not exist to induce the actual module into the error state for testing.
3. **Algorithm Testing** – Algorithm testing utilizing the defined ports, interfaces, and services that a module provides is the preferred method. This method most clearly meets the requirements of FIPS 140-3 Implementation Guidance (IG) 2.3.A. If this preferred method is not possible where the module's defined set of ports, interfaces, and services does not allow access to internal algorithmic engines, two alternative methods may be utilized:
    a. A module may be modified under the supervision of the Cryptographic and Security Testing Laboratory (CSTL) for testing purposes to allow access to the algorithmic engines (e.g., test jig, test API), or
    b. A module simulator may be utilized.

3.3.2.3 Harness

No clearly articulated interpretation of when and how a test harness can and should be used is available, as much of what is known comes from experienced vendors who developed specialized test harnesses around their respective modules and within the restricted operating environments.

3.3.2.4 Manual

No clearly articulated interpretation of when and how a manual process can and should be used is available, as much of what is known comes from the need for human interaction to trigger events or an inability to trigger the steps in an automated approach.

3.3.2.5 Other

As noted earlier, due to the diversity and complexity of operating environments, the toolset needed to perform the gathering of relevant TE may not fit precisely within the above five test methods. Therefore, there is a need for a catch-all method that enables the tester to comprehensively describe the methodology used to capture the TE. The testing techniques can evolve over time and emergent testing techniques can be added to this list.

3.3.3 Test Methods Allowed

Table 4 maps the allowed test methods to the grouping of associated TEs for the purpose of condensing the resulting table.

Table 4 Allowed Test Methods

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
02.12.01	X	X	X	X	✓	✓
02.13.03	X	X	X	✓	X	✓
02.15.03	X	X	X	X	✓	✓
02.15.05, 02.16.04, 02.17.04	✓	X	X	X	✓	✓
02.16.02, 02.17.02	X	X	X	✓	X	✓
02.19.02	✓	X	X	✓	✓	✓
02.22.02	✓	X	X	✓	X	✓
02.24.02	✓	X	X	✓	✓	✓
02.26.03, 02.26.04, 02.26.05, 02.28.01, 02.28.02, 02.30.02	✓	X	X	✓	X	✓
03.01.04, 03.02.01, 03.14.03, 03.15.03,	✓	X	X	✓	✓	✓

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
03.15.04, 03.15.06						
03.05.01, 03.05.02	✓	X	X	✓	✓	✓
03.06.01, 03.06.02, 03.07.01, 03.07.02, 03.07.04, 03.07.08	✓	X	X	✓	✓	✓
03.08.01, 03.08.02	✓	✓	X	✓	✓	✓
03.09.02, 03.10.02, 03.10.04	✓	✓	X	✓	✓	✓
03.11.01, 03.11.03	✓	X	X	✓	✓	✓
03.13.02	X	X	X	X	✓	✓
03.18.02, 03.19.02, 03.19.04, 03.20.01, 03.21.01	✓	X	X	✓	✓	✓
03.22.01	✓	X	X	✓	✓	✓
04.02.02, 04.02.03	✓	X	X	✓	✓	✓
04.07.03	✓	X	X	✓	✓	✓
04.11.02	✓	X	X	✓	✓	✓
04.13.01, 04.13.02, 04.13.03	✓	✓	✓	✓	✓	✓
04.14.02	✓	X	X	✓	✓	✓

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
04.15.01	✓	X	X	✓	✓	✓
04.18.01, 04.19.02, 04.19.03, 04.20.01, 04.20.03, 04.21.02, 04.22.02	✓	X	X	✓	✓	✓
04.23.01, 04.25.01, 04.25.02, 04.25.03	✓	X	X	✓	✓	✓
04.28.01, 04.29.01, 04.32.01, 04.33.01, 04.34.01, 04.35.02, 05.13.08	✓	✓	✓	✓	✓	✓
04.37.02, 04.38.02	✓	X	X	✓	✓	✓
04.39.02, 04.39.03, 04.39.04, 04.42.03, 04.42.04	✓	X	X	✓	✓	✓
04.43.02, 04.44.02	✓	X	X	✓	✓	✓
04.45.02, 04.45.03, 04.47.01, 04.48.01, 04.52.01, 04.54.02, 04.54.03, 04.55.02	✓	X	X	✓	✓	✓

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
04.53.01	✓	✓	✓	✓	✓	✓
04.56.02	✓	X	X	✓	✓	✓
04.59.01	✓	X	X	✓	✓	✓
05.05.05	✓	✓	✓	✓	✓	✓
05.05.07, 05.06.06, 05.08.01, 05.08.02, 05.11.01, 05.11.02, 05.12.02, 05.13.03, 05.13.04, 05.13.05	✓	X	X	✓	✓	✓
05.06.02	✓	✓	✓	✓	✓	✓
05.06.03	✓	X	X	✓	✓	✓
05.06.04	✓	X	X	✓	✓	✓
05.13.01, 05.13.02	✓	X	X	✓	✓	✓
05.13.06	✓	X	X	✓	✓	✓
05.15.01, 05.15.02, 05.16.03, 05.17.02	✓	X	X	✓	✓	✓
05.20.01	✓	✓	✓	✓	✓	✓
05.23.01	✓	✓	✓	✓	✓	✓
06.05.01, 06.05.02, 06.05.03, 06.06.01,	✓	✓	✓	✓	✓	✓

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
06.06.02, 06.08.01, 06.08.03						
06.06.02, 06.08.03	✓	✓	✓	✓	✓	✓
09.01.02, 09.01.03, 09.02.02, 09.03.02, 09.03.03, 09.14.02, 09.16.03, 09.25.02, 09.27.02	✓	X	X	✓	✓	✓
09.21.02, 09.21.03, 09.21.04, 09.22.01	✓	X	X	✓	✓	✓
09.24.02	✓	X	X	✓	✓	✓
09.28.02, 09.28.03, 09.28.04	✓	X	X	✓	✓	✓
09.33.02	✓	X	X	✓	✓	✓
09.36.02, 09.37.02	✓	X	X	✓	✓	✓
10.07.03, 10.08.03, 10.09.03, 10.10.01, 10.10.02, 10.28.02	✓	X	X	✓	✓	✓
10.07.04	✓	X	X	✓	✓	✓

TE (TE##.##.##	Debugger	Simulator	Emulator	Harness	Manual	Other
10.25.02, 10.27.01	✓	X	X	✓	✓	✓
10.35.04	✓	✓	X	✓	✓	✓
10.53.02, 10.53.03	✓	X	X	✓	✓	✓
11.08.06, 11.08.09, 11.11.01	✓	X	X	✓	✓	✓
11.13.02	✓	X	X	✓	✓	✓
11.28.02, 11.28.03, 11.28.04	✓	✓	✓	✓	✓	✓
11.32.02	✓	X	X	✓	✓	✓

3.4 Improvement of TE Filtering Coverage

TE filters serve as a pivotal mechanism to streamline the classification and evaluation of TE, ensuring that only relevant and applicable tests are conducted based on specific module characteristics. A proper set of applicable TEs tailored by a given module specification refines the required assessments and optimizes the validation process.

With the growing complexity of cryptographic modules and the need for efficient validation, TE filters are designed to:

- Target specific needs through focusing on applicable tests by narrowing down evidence requirements based on module attributes such as type, security level, and operational environment
- Reduce redundancy through minimizing repetitive validation steps by filtering out TEs that are not relevant to a given module's configuration or features
- Enhance automation through supporting automated workflows by integrating filters into structured JSON schemas, aligning with automation tools like WebCryptik

This document delves into the methodologies and criteria for applying TE filters, the implementation of filtering mechanisms, and their role in achieving a more efficient and scalable CMVP. By leveraging these filters, vendors and validators can focus on precise compliance requirements, reducing manual overhead while maintaining robust security standards.

827 Table 5 is excerpted from ISO/IEC 19790:2012, which is the base of FIPS 140-3. It provides a structured
828 summary of the FIPS 140-3 security requirements across various requirement areas. It outlines the
829 security levels applicable to each category, specifying the testing expectations and security assurances
830 needed to meet compliance. The table serves as a reference for understanding how different
831 cryptographic module components must align with FIPS 140-3 standards, ensuring consistent evaluation
832 and validation. Each requirement area focuses on distinct security aspects, such as module
833 specifications, authentication mechanisms, physical security, and lifecycle assurance, enabling a
834 comprehensive approach to cryptographic module validation.

835 **Table 5 Summary of FIPS 140-3 Security Requirements**

Requirement Area		FIPS 140-3 Security Level			
		1	2	3	4
1	**General**	No security testing requirements (i.e., no TEs)			
2	**Cryptographic Module Specification**	Specification of cryptographic module, cryptographic boundary, approved security functions, and normal and degraded modes of operation. Description of cryptographic module, including all hardware, software, and firmware components. All services provide status information to indicate when the service utilizes an approved cryptographic algorithm, security function, or process in an approved manner.			
3	**Cryptographic Module Interfaces**	Required and optional interfaces. Specification of all interfaces and of all input and output data paths		Trusted channel	
4	**Roles, Services, and Authentication**	Logical separation of required and optional roles and services	Role-based or identity-based operator authentication	Identity-based operator authentication	Multi-factor authentication
5	**Software / Firmware Security**	Approved integrity technique. Defined SFMI, HFMI, and HSMI. Executable code	Approved digital signature or keyed message authentication code-based integrity test	Approved digital signature-based integrity test	
6	**Operational Environment**	Non-modifiable. Limited or Modifiable Control of SSPs	Modifiable. Role-based or discretionary access control. Audit mechanism		
7	**Physical Security**	Production-grade components	Tamper evidence. Opaque covering	Tamper detection and response for covers and doors.	Tamper detection and re-

Requirement Area		FIPS 140-3 Security Level			
		1	2	3	4
			or enclosure	Strong enclosure or coating. Protection from direct probing EFP or EFT	sponse envelope. EFP. Fault injection mitigation
8	**Non-Invasive Security**	Module is designed to mitigate against non-invasive attacks specified in Annex "F"			
		Documentation and effectiveness of mitigation techniques specified in Annex "F"		Mitigation testing	Mitigation testing
9	**Security Parameter Management**	Random bit generators, SSP generation, establishment, entry & output, storage & zeroization			
		Automated SSP transport or SSP agreement using approved methods			
		Manually established SSPs may be entered or output in plaintext form		Manually established SSPs may be entered or output in either encrypted form, via a trusted channel, or using split knowledge procedures	
10	**Self-Tests**	Pre-operational: software/firmware integrity, bypass, and critical functions test			
		Conditional: cryptographic algorithm, pair-wise consistency, SW/FW loading, manual entry, conditional bypass & critical functions test			
11	**Life-Cycle Assurance**				
	Configuration Management	Configuration management system for cryptographic module, components, and documentation. Each is uniquely identified and tracked throughout its lifecycle		Automated configuration management system	
	Design	Module designed to allow testing of all provided security-related services			
	FSM	Finite State Model			

Requirement Area		FIPS 140-3 Security Level			
		1	2	3	4
	Development	Annotated source code, schematics, or HDL	Software high-level language and hardware high-level descriptive language		Documentation annotated with pre-conditions upon entry into module components and post-conditions expected to be true when components is completed
	Testing	Functional testing		Low-level testing	
	Delivery & Operation	Initialization procedures	Delivery procedures		Operator authentication using vendor-provided authentication information
	Guidance	Administrator and non-administrator guidance			
12	**Mitigation of Other Attacks**	Specification of mitigation of attacks for which no testable requirements are currently available			Specification of mitigation of attacks with testable requirements

Building on the summary of FIPS 140-3 security requirements in Table 5, Table 6 provides a more granular analysis of the number of security requirements per ISO/IEC 24759:2014(2015), which is a companion document to ISO/IEC 19790 specifying the derived test requirements, across different implementation areas. This table categorizes security requirements based on the module's type being Software (SW), Firmware (FW), Hardware (HW), SW-HW hybrid (SW-H), or FW-HW hybrid (FW-H), and further differentiates them by security levels. The breakdown facilitates a clearer understanding of the distribution of TE requirements, highlighting how various module implementations align with compliance expectations at each level.

The number of total TEs and the percentage of applicable TEs will indicate how many TEs are not applicable. By filtering out these non-applicable TEs with public consensus, the CSTL can more directly perform the required testing.

847 **Table 6 An overview of the number of Security Requirements**

Area	To-tal TEs	Security Level 1					Security Level 2					Security Level 3					Security Level 4				
		SW	FW	HW	SW -H	FW -H	SW	FW	HW	SW -H	FW -H	SW	FW	HW	SW -H	FW -H	SW	FW	HW	SW -H	FW -H
2	65	40	45	49	55	60	40	45	49	55	60	40	45	49	55	60	40	45	49	55	60
3	53	41	43	43	43	43	41	43	43	43	43	46	48	52	52	52	47	49	53	53	53
4	74	45	45	45	45	45	63	63	63	63	63	70	70	70	70	70	71	71	71	71	71
5	39	23	23	23	30	30	30	30	29	37	37	32	32	30	39	39	32	32	30	39	39
6	50	10	10	10	10	10	50	50	50	50	50	0	0	0	0	0	0	0	0	0	0
7	82	0	14	14	14	14	0	27	27	27	27	0	69	69	69	69	0	78	78	78	78
8	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4
9	63	44	43	43	44	43	48	47	47	48	47	56	56	56	56	56	57	57	57	57	57
10	74	68	68	68	68	68	68	68	68	68	68	74	74	74	74	74	74	74	74	74	74
11	52	36	36	35	38	38	41	41	41	44	44	44	44	44	47	47	49	49	49	52	52
12	5	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	5	5	5	5	5
A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Total TEs	567	317	337	340	357	361	391	424	427	445	449	373	449	455	473	478	384	469	475	493	498
% Appli-cable	100	56	59	60	63	64	69	75	75	78	79	66	79	80	83	84	68	83	84	87	88

848 We recognize that software implementations only support levels 1 and 2. However:

849 • The Area 2 TEs include requirements from security level 1 through level 4, which are listed in
850 Table 7. This area's requirements are about the Cryptographic Module Specification and are the
851 same for all four security levels. The unified area 2 requirements are reflected by the numbers
852 of TEs in the red rectangle boxes on Table 6.

853 • The Area 7 TEs include requirements from security level 1 through level 4, which are listed in
854 Table 6. The Physical Security requirements in Area 7 are incremental for cryptographic modules

from a low security level to a higher level. The numbers of TEs in the green rectangle boxes on Table 6 illustrate this trend.

Table 6 and Table 7 in section 3.4.2 serve as examples of how the basic TE filters work by listing all applicable TEs and non-applicable TEs for a given type of module at any possible security level. A complete set of TE tables elaborating on Table 6 is provided in Appendix B of this document.

3.4.1 TE Filtering Criteria

The TE Filtering criteria consists of the Module Information and Supplemental Information from the WebCryptik as the base. The CMVP provided Module Supplemental Information (dated 2025-06-25), but this is not currently used to tailor the set of TEs to fit the module under test.

In the CMVP's Module Supplemental Information (MSI) document, most Supplemental Information questions map to the security assertions (AS), test requirements (TE), implementation guidance (IG), and security policies (SP), but a few questions are not mapped to any of these and are left blank. The list below reflects the CMVP's current MSI document. The TE Workstream provides a complete mapping of MSI questions to relevant TEs in Table 8.

By reviewing all TEs contained in the WebCryptik Br1 v1.0.6, the TE Workstream completed the list of criteria, including the basic filters and supplemental filters, as follows:

- **Basic Filters**
  - Module Embodiment: Single Chip, Multi-Chip Embedded, Multi-Chip Standalone
  - Module Type: Software, Hardware, Firmware, Software-hybrid, Firmware-hybrid
  - Operational Environment: modifiable, limited, non-modifiable
  - Section Level: Per Table 4, area 6 is not applicable to Level 3 and Level 4
- **Supplemental Filters**
  - **Cryptographic module specification**
    - Does the module implement OTAR? – IG D.C
    - Does the module have a non-approved mode? – IG 2.4.A
    - Does the module require initialization steps to operate in the approved mode? – Certificate Caveat and SP
    - Does the module have excluded components? – AS02.13, AS02.14
    - Does the module allow a degraded mode of operation? – AS02.25
    - Does the module have an implementation of PAA or PAI? – IG 2.3.C
    - Does the module contain an embedded or have a bound cryptographic module? – IG 2.3.A
    - Does the module have any critical functions? – AS10.16, AS10.23, AS10.24, AS10.52
    - Is the module a sub-chip implementation? – IG 2.3.B

- Does the module's approved mode make use of any non-approved algorithm? – IG 2.4.A
- Does the module have a non-compliant state?

- **Cryptographic module interfaces**
  - Does the module receive any of its input from an external input device? – TE03.05.02, TE03.06.02, TE03.08.02, TE03.11.02
  - Does the module provide any of its output through an external output device? – TE03.05.02, TE03.06.02, TE03.08.02, TE03.11.02
  - Does the module implement a Trusted Channel? – IG 3.4.A
  - Is there a control output interface? – AS03.09, AS03.10
- **Roles, services, and authentication**
  - Does the module support concurrent operators? – AS04.02
  - Does the module support any authentication mechanism? – AS04.43-AS04.55
  - Does the module use identity-based authentication?
  - Does the module support role-based authentication?
  - Does the module support multi-factor-based authentication? – AS03.22
  - Does the module have a bypass capability? – AS04.22, AS10.21-AS10.22, AS10.47-AS10.51
  - Is there a maintenance role? – AS04.07
  - Is there a user role? – AS04.06
  - Can operators change roles? – AS04.38, AS04.42
  - Does the module support self-initiated cryptographic output? – AS04.23-AS04.26
  - Is default information used for first-time authentication? – AS04.46
  - Does the module support software/firmware loading? – AS04.28-AS04.33, AS05.13
  - Is a complete image replacement supported within software/firmware loading? – AS04.33-AS04.35
- **Software/firmware security**
  - Does the module use a hash or MAC to verify the integrity of its software/firmware? – TE05.05.03
  - Does the module use a digital signature to verify the integrity of its software/firmware? – TE05.05.04
  - Does the module use an EDC for the software/firmware components of a hardware module? – AS05.06
  - Does the module contain any non-reconfigurable memory? – IG 5.A

926 - Does the module utilize open-source software? – Annex B
927 - **Operational environment**
928 - None
929 - **Physical security**
930 - Is there a maintenance access interface? – AS07.11-AS07.13, TE11.08.07
931 - Are there any ventilation holes or slits? – AS07.20, AS07.25
932 - Are there any removable covers/doors? – AS07.22, TE07.39.02, TE07.39.05,
933 AS07.47, TE07.51.02, TE07.51.07, TE07.51.08, AS07.62, TE07.65.02, TE07.65.07,
934 TE07.65.08
935 - Are there tamper seals? – IG 7.3.A
936 - Are there tamper seals applied by the module user?
937 - Does the module implement EFP or EFT mechanisms?
938 - **Non-invasive security**
939 - None
940 - **Sensitive security parameters management**
941 - Does the module support input and/or output of SSPs or other sensitive data? –
942 AS09.13, AS09.18, AS09.19
943 - Are there plaintext keys, CSPs, or sensitive data output? – AS09.16-
944 AS09.17
945 - Does the module support manual/direct entry of SSPs? AS09.15,
946 AS10.42-AS10.46, TE10.46.04
947 - Is split knowledge utilized? – AS09.21, AS09.22, AS09.23
948 - Is one-time programmable (OTP) memory used in the module? – IG 9.7.A
949 - **Self-tests**
950 - None
951 - **Life-cycle assurance**
952 - Are there any CVEs related to this module? – IG 11.A
953 - **Mitigation of other attacks**
954 - Is the module designed to mitigate other attacks?
955 - **Approved security functions**
956 - Are any non-NIST curves used? – IG C.A

957 3.4.2 TEs Impacted by Basic TE Filters

958 To ensure a structured approach to TE filtering, it is necessary to categorize TEs based on the security
959 level and module type. Table 6 presents a detailed breakdown of the TEs applicable to different security
960 levels for software modules, illustrating how filtering criteria refine the validation scope. By segmenting

961 TEs according to security requirements, this table helps streamline the testing process, ensuring that
962 only the relevant test evidence is considered for a given module configuration. This targeted approach
963 enhances efficiency while maintaining rigorous security standards.

964 We recognize that software implementations only support levels 1 and 2. However, Table 7 lists the
965 Area 2 Cryptographic Module Specification TEs required from security level 1 through level 4, and Table
966 8 lists the Area 7 Physical Security TEs for all four security levels.

967 **Table 7 Area 2 TEs Filtered by Security Level for Software Modules**

Sec Level	Applicable TEs	Non-Applicable TEs	TEs N/A due to Module Type
1	TE02.03.01, TE02.03.02, TE02.07.01, TE02.07.02, TE02.09.01, TE02.10.01, TE02.10.02, TE02.11.01, TE02.11.02, TE02.12.01, TE02.13.01, TE02.13.02, TE02.13.03, TE02.14.01, TE02.16.01, TE02.16.02, TE02.16.03, TE02.16.04, TE02.16.05, TE02.19.01, TE02.19.02, TE02.20.01, TE02.20.02, TE02.20.03, TE02.20.04, TE02.21.01, TE02.21.02, TE02.22.01, TE02.22.02, TE02.24.01, TE02.24.02, TE02.26.01, TE02.26.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.01, TE02.30.02	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01
2	TE02.03.01, TE02.03.02, TE02.07.01, TE02.07.02, TE02.09.01, TE02.10.01, TE02.10.02, TE02.11.01, TE02.11.02, TE02.12.01, TE02.13.01, TE02.13.02, TE02.13.03, TE02.14.01, TE02.16.01, TE02.16.02, TE02.16.03, TE02.16.04, TE02.16.05, TE02.19.01, TE02.19.02, TE02.20.01, TE02.20.02, TE02.20.03, TE02.20.04, TE02.21.01, TE02.21.02, TE02.22.01, TE02.22.02, TE02.24.01, TE02.24.02, TE02.26.01, TE02.26.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.01, TE02.30.02	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01

Sec Level	Applicable TEs	Non-Applicable TEs	TEs N/A due to Module Type
3	TE02.03.01, TE02.03.02, TE02.07.01, TE02.07.02, TE02.09.01, TE02.10.01, TE02.10.02, TE02.11.01, TE02.11.02, TE02.12.01, TE02.13.01, TE02.13.02, TE02.13.03, TE02.14.01, TE02.16.01, TE02.16.02, TE02.16.03, TE02.16.04, TE02.16.05, TE02.19.01, TE02.19.02, TE02.20.01, TE02.20.02, TE02.20.03, TE02.20.04, TE02.21.01, TE02.21.02, TE02.22.01, TE02.22.02, TE02.24.01, TE02.24.02, TE02.26.01, TE02.26.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.01, TE02.30.02	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01
4	TE02.03.01, TE02.03.02, TE02.07.01, TE02.07.02, TE02.09.01, TE02.10.01, TE02.10.02, TE02.11.01, TE02.11.02, TE02.12.01, TE02.13.01, TE02.13.02, TE02.13.03, TE02.14.01, TE02.16.01, TE02.16.02, TE02.16.03, TE02.16.04, TE02.16.05, TE02.19.01, TE02.19.02, TE02.20.01, TE02.20.02, TE02.20.03, TE02.20.04, TE02.21.01, TE02.21.02, TE02.22.01, TE02.22.02, TE02.24.01, TE02.24.02, TE02.26.01, TE02.26.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.01, TE02.30.02	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01	TE02.15.01, TE02.15.02, TE02.15.03, TE02.15.04, TE02.15.05, TE02.15.06, TE02.15.07, TE02.15.08, TE02.15.09, TE02.15.10, TE02.15.11, TE02.15.12, TE02.15.13, TE02.15.14, TE02.17.01, TE02.17.02, TE02.17.03, TE02.17.04, TE02.17.05, TE02.17.06, TE02.17.07, TE02.17.08, TE02.17.09, TE02.17.10, TE02.18.01

While Table 7 focuses on the impact of TE filtering for software modules, the filtering criteria must also be applied to hardware-based implementations. Table 8 extends this analysis by examining TEs specific to single-chip hardware modules, mapping the applicable security requirements to different security levels. This comparison highlights the distinctions in validation approaches between software and hardware modules, ensuring that the filtering process remains consistent and comprehensive across various module types.

Table 8 Area 7 TEs Filtered by Security Level for Single Chip Hardware Modules

Sec Level	Applicable TEs	Non-Applicable TEs	TEs N/A due to Module Type/Embodiment
1	TE07.01.01, TE07.01.02, TE07.09.01, TE07.09.02, TE07.10.01, TE07.10.02, TE07.11.01, TE07.11.02,	TE07.19.01, TE07.20.01, TE07.25.01, TE07.26.01, TE07.26.02, TE07.27.01, TE07.32.01, TE07.33.01, TE07.35.01, TE07.37.01, TE07.37.02, TE07.37.03,	TE07.43.01, TE07.60.01

Sec Level	Applicable TEs	Non-Applicable TEs	TEs N/A due to Module Type/Embodiment
	TE07.12.01, TE07.13.01, TE07.15.01, TE07.15.02	TE07.39.01, TE07.39.02, TE07.39.03, TE07.39.04, TE07.39.05, TE07.39.06, TE07.41.01, TE07.41.02, TE07.42.01, TE07.42.02, TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.53.01, TE07.55.01, TE07.57.01, TE07.58.01, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09, TE07.67.01, TE07.71.01, TE07.71.02, TE07.73.01, TE07.77.01, TE07.77.02, TE07.77.03, TE07.77.04, TE07.81.01, TE07.81.02, TE07.81.03	
2	TE07.01.01, TE07.01.02, TE07.09.01, TE07.09.02, TE07.10.01, TE07.10.02, TE07.11.01, TE07.11.02, TE07.12.01, TE07.13.01, TE07.15.01, TE07.15.02, TE07.19.01, TE07.20.01, TE07.35.01	TE07.25.01, TE07.26.01, TE07.26.02, TE07.27.01, TE07.32.01, TE07.33.01, TE07.37.01, TE07.37.02, TE07.37.03, TE07.39.01, TE07.39.02, TE07.39.03, TE07.39.04, TE07.39.05, TE07.39.06, TE07.41.01, TE07.41.02, TE07.42.01, TE07.42.02, TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.53.01, TE07.55.01, TE07.57.01, TE07.58.01, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09, TE07.67.01, TE07.71.01, TE07.71.02, TE07.73.01, TE07.77.01, TE07.77.02, TE07.77.03, TE07.77.04, TE07.81.01, TE07.81.02, TE07.81.03	TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.60.01, TE07.62.01, TE07.63.01
3	TE07.01.01, TE07.01.02, TE07.09.01, TE07.09.02, TE07.10.01, TE07.10.02, TE07.11.01, TE07.11.02, TE07.12.01, TE07.13.01, TE07.15.01, TE07.15.02, TE07.19.01, TE07.20.01,	TE07.32.01, TE07.33.01, TE07.41.01, TE07.41.02, TE07.42.01, TE07.42.02, TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03,	TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02,

Sec Level	Applicable TEs	Non-Applicable TEs	TEs N/A due to Module Type/Embodiment
	TE07.25.01, TE07.26.01, TE07.26.02, TE07.27.01, TE07.35.01, TE07.37.01, TE07.37.02, TE07.37.03, TE07.39.01, TE07.39.02, TE07.39.03, TE07.39.04, TE07.39.05, TE07.39.06, TE07.73.01, TE07.77.01, TE07.77.02, TE07.77.03, TE07.77.04, TE07.81.01, TE07.81.02, TE07.81.03	TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.53.01, TE07.55.01, TE07.57.01, TE07.58.01, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09, TE07.67.01, TE07.71.01, TE07.71.02	TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09
4	TE07.01.01, TE07.01.02, TE07.09.01, TE07.09.02, TE07.10.01, TE07.10.02, TE07.11.01, TE07.11.02, TE07.12.01, TE07.13.01, TE07.15.01, TE07.15.02, TE07.19.01, TE07.20.01, TE07.25.01, TE07.26.01, TE07.26.02, TE07.27.01, TE07.32.01, TE07.33.01, TE07.35.01, TE07.37.01, TE07.37.02, TE07.37.03, TE07.39.01, TE07.39.02, TE07.39.03, TE07.39.04, TE07.39.05, TE07.39.06, TE07.41.01, TE07.41.02, TE07.42.01, TE07.42.02, TE07.77.01, TE07.77.02, TE07.77.03, TE07.77.04	TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.53.01, TE07.55.01, TE07.57.01, TE07.58.01, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09, TE07.67.01, TE07.71.01, TE07.71.02, TE07.73.01, TE07.81.01, TE07.81.02, TE07.81.03	TE07.43.01, TE07.44.01, TE07.45.01, TE07.45.02, TE07.46.01, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.50.01, TE07.50.02, TE07.50.03, TE07.51.01, TE07.51.02, TE07.51.03, TE07.51.04, TE07.51.05, TE07.51.06, TE07.51.07, TE07.51.08, TE07.51.09, TE07.53.01, TE07.55.01, TE07.57.01, TE07.58.01, TE07.60.01, TE07.62.01, TE07.63.01, TE07.65.01, TE07.65.02, TE07.65.03, TE07.65.04, TE07.65.05, TE07.65.06, TE07.65.07, TE07.65.08, TE07.65.09, TE07.67.01, TE07.71.01, TE07.71.02

3.4.3 TE Impacted by Supplemental TE Filters

In addition to the basic TE filtering criteria, supplemental filters further refine the selection of applicable test evidence based on specific module properties and security features. Table 9 highlights the TEs affected by these supplemental filtering properties, which include factors such as authentication mechanisms, cryptographic output capabilities, tamper response measures, and other specialized security attributes. By applying these filters, the validation process can be optimized to focus on the most relevant security assurances while reducing redundant or inapplicable tests. This targeted approach enhances the efficiency and accuracy of the TE selection process.

983 **Table 9 TEs Affected by the Supplemental Filtering Properties**

Filter Property	Include If True	Exclude If False	Number of Affected TEs
Has Excluded Components		TE02.13.01, TE02.13.02, TE02.13.03, TE02.14.01, TE02.15.05, TE02.16.04, TE02.17.04	7
Has EFP		TE07.77.01, TE07.77.02, TE07.77.03, TE07.77.04	4
Uses Split Knowledge		TE09.21.01, TE09.21.02, TE09.21.03, TE09.21.04, TE09.22.01, TE09.23.01, TE09.23.02, TE09.23.04, TE09.24.01	9
Allows Self-Initiated Cryptographic Output		TE04.23.01, TE04.25.01, TE04.25.02, TE04.25.03	4
Supports Bypass Capability		TE04.18.01, TE04.19.01, TE04.19.02, TE04.19.03, TE04.20.01, TE04.20.02, TE04.20.03, TE04.21.01, TE04.21.02, TE04.22.01, TE04.22.02, TE10.21.01, TE10.21.02, TE10.21.03, TE10.21.04, TE10.22.01, TE10.22.02, TE10.22.03, TE10.22.04, TE10.22.05, TE10.48.01, TE10.48.02, TE10.48.03, TE10.49.01, TE10.49.02, TE10.49.03, TE10.51.01, TE10.51.02, TE10.51.03	29
Has Identity-Based Authentication		TE03.20.01, TE04.39.01, TE04.39.02, TE04.39.03, TE04.39.04, TE04.42.01, TE04.42.02, TE04.42.03, TE04.42.04, TE09.22.01	10
Provides Maintenance Access Interface	TE07.50.03	TE07.11.01, TE07.11.02, TE07.12.01, TE07.13.01, TE07.51.07, TE07.51.08, TE07.65.02, TE07.65.07, TE07.65.08, TE11.08.07	11
Uses EDC		TE05.06.02, TE05.07.01	2
Supports Manual SSP Entry		TE09.14.01, TE09.14.02, TE10.46.01, TE10.46.02, TE10.46.03, TE10.46.04	6
Supports Concurrent Operators		TE04.02.01, TE04.02.02, TE04.02.03	3
Supports Software Firmware Loading		TE04.28.01, TE04.29.01, TE04.32.01, TE04.34.01, TE05.13.01, TE05.13.02, TE05.13.03, TE05.13.04, TE05.13.05, TE05.13.06, TE05.13.07, TE05.13.08	12

Filter Property	Include If True	Exclude If False	Number of Affected TEs
Supports Complete Image Replacement		TE04.33.01, TE04.35.01, TE04.35.02	3
Uses Hash MAC Integrity		TE05.05.03	1
Has Control Output		TE03.09.01, TE03.09.02, TE03.10.01, TE03.10.02, TE03.10.03, TE03.10.04, TE03.10.05	7
Has Ventilation or Slits		TE07.20.01, TE07.25.01	2
Has EDC		TE10.46.02, TE10.46.03	2
Has External Input Device		TE03.05.02, TE03.08.02	2
Has User Role		TE04.06.01	1
Has External Output Device		TE03.06.02, TE03.11.02	2
Has Removable Cover	TE07.50.03	TE07.13.01, TE07.20.01, TE07.25.01, TE07.39.02, TE07.39.05, TE07.47.01, TE07.47.02, TE07.48.01, TE07.48.02, TE07.51.02, TE07.51.07, TE07.51.08, TE07.62.01, TE07.63.01, TE07.65.02, TE07.65.07, TE07.65.08	18
Outputs Sensitive Data as Plaintext		TE09.16.01, TE09.16.02, TE09.16.03	3
Has Critical Functions		TE10.24.01, TE10.24.02	2
Uses Authentication		TE04.43.01, TE04.43.02, TE04.44.01, TE04.44.02, TE04.45.01, TE04.45.02, TE04.45.03, TE04.47.01, TE04.48.01, TE04.50.01, TE04.50.02, TE04.51.01, TE04.51.02, TE04.52.01, TE04.53.01, TE04.54.01, TE04.54.02, TE04.54.03, TE04.55.01, TE04.55.02	20
Uses Role-Based Authentication		TE04.37.01, TE04.37.02, TE04.38.01, TE04.38.02	4

Filter Property	Include If True	Exclude If False	Number of Affected TEs
Has Default Authentication Data		TE04.45.03	1
Has Degraded Mode		TE02.26.01, TE02.26.02, TE02.26.03, TE02.26.04, TE02.26.05, TE02.28.01, TE02.28.02, TE02.30.01, TE02.30.02	9
Has EFT		TE07.81.01, TE07.81.02, TE07.81.03	3
Has Trusted Channel		TE03.16.01, TE03.18.01, TE03.18.02, TE03.19.01, TE03.19.02, TE03.19.03, TE03.19.04, TE03.20.01, TE03.21.01, TE03.22.01, TE09.21.01, TE09.21.04	12
Uses Multi-Factor Authentication		TE04.59.01, TE09.24.01, TE09.24.02	3
Allows Operator to Change Roles		TE04.38.01, TE04.38.02, TE04.42.01, TE04.42.02, TE04.42.03, TE04.42.04	6
Uses Digital Signature Integrity		TE05.05.04	1
Has Maintenance Role		TE04.07.01, TE04.07.02, TE04.07.03	3
Has Additional Mitigations		TE12.01.01, TE12.02.01, TE12.04.01, TE12.04.02, TE12.04.03	5
Supports Sensitive Data I/O		TE09.13.01, TE09.13.02, TE09.13.03, TE09.18.01, TE09.18.02, TE09.19.01	6
Has Tamper Seals		TE07.27.01, TE07.48.01, TE07.48.02, TE07.63.01	4
Has CVE		TE11.38.03	1
Total number of TEs affected by the supplemental filter properties			**192**

Note: The total number of the TEs affected by the supplemental filter properties is not the sum of the numbers in the column of "Number of Affected TEs" (i.e., 218) because some TEs are affected by multiple filter properties and so appear multiple times in Table 9.

3.5 Removing ASes not separately tested

Some assertions (ASes) are “not separately tested”, as indicated in ISO/IEC 24759 [3]and they do not have associated TEs.

These ASes depend on the completion of other ASes and their TEs. For example, **AS05.22** is not separately tested but is instead tested as part of **AS05.05**. Table 10 highlights ASes that are not separately tested. Since these ASes are conditional in nature, a solution to the problem of these assertions that could be utilized is to use these assertions to further automate the report writing process. In this instance, the AS that is not separately tested could be omitted from the report template provided by the NCCoE ACMVP server if the server will include ASes in addition to TEs.

The TE Workstream does not address the dependency at the TE level (e.g., TE10.28.02 and TE10.34.03) as opposed to the AS level.

Table 10 Assertions (ASes) not separately tested

FIPS 140-3 Section Title	ASes not separately tested
General	N/A
Cryptographic Module Specification	AS02.01, AS02.02, AS02.04, AS02.05, AS02.06, AS02.08, AS02.25, AS02.26, AS02.29, AS02.31, AS02.32
Cryptographic Module Interfaces	AS03.12, AS03.17
Roles, Services, and Authentication	AS04.01, AS04.05, AS04.08, AS04.09, AS04.10, AS04.12, AS04.16, AS04.17, AS04.24, AS04.26, AS04.27, AS04.30, AS04.31, AS04.36, AS04.40, AS04.41, AS04.46, AS04.49, AS04.57, AS04.58
Software/Firmware Security	AS05.01, AS05.03, AS05.09, AS05.10, AS05.14, AS05.18, AS05.19, AS05.21, AS05.22
Operational Environment	AS06.01, AS06.02, AS06.04, AS06.09, AS06.16, AS06.21, AS06.22, AS06.23, AS06.29
Physical Security	AS07.02, AS07.03, AS07.04, AS07.05, AS07.06, AS07.07, AS07.08, AS07.14, AS07.16, AS07.17, AS07.18, AS07.21, AS07.22, AS07.23, AS07.24, AS07.28, AS07.29, AS07.30, AS07.31, AS07.34, AS07.36, AS07.38, AS07.40, AS07.49, AS07.52, AS07.54, AS07.56, AS07.59, AS07.61, AS07.64, AS07.66, AS07.68, AS07.69, AS07.70, AS07.72, AS07.74, AS07.75, AS07.76, AS07.78, AS07.79, AS07.80, AS07.81, AS07.82, AS07.83, AS07.84, AS07.85, AS07.86
Non-Invasive Security	N/A
Sensitive Security Parameter Management	AS09.11, AS09.12, AS09.15, AS09.17, AS09.20, AS09.26, AS09.30, AS09.34, AS09.35

FIPS 140-3 Section Title	ASes not separately tested
Self-Tests	AS10.01, AS10.02, AS10.03, AS10.04, AS10.05, AS10.06, AS10.13, AS10.14, AS10.16, AS10.17, AS10.18, AS10.19, AS10.23, AS10.26, AS10.30, AS10.31, AS10.32, AS10.32, AS10.36, AS10.38, AS10.39, AS10.40, AS10.41, AS10.42, AS10.43, AS10.44, AS10.45, AS10.47, AS10.50, AS10.52, AS10.55
Life-Cycle Assurance	AS11.02, AS11.07, AS11.09, AS11.10, AS11.12, AS11.14, AS11.20, AS11.22, AS11.27
Mitigation of Other Attacks	None

999

4 Protocol Workstream

The Protocol Workstream defines the interactions between the automated CMVP server and the ACMVP clients supporting a proof-of-concept of automation capabilities. The CMVP hosts a demonstration server for interoperability and testing purposes through NIST. This is referred to as the ACMVP Demo Server covered in section 4.5.

4.1 Protocol Workstream Collaborators

The ACMVP Protocol Workstream is led by Barry Fussell and Andrew Karcher of Cisco and Christopher Celi of NIST, with contributions from Panos Kampanakis of Amazon, Michael McCarl and Deborah Harrington of AEGISOLVE, Alex Thurston of Lightship, Stephan Mueller and Walker Riley of atsec information security, Mike Grimm of Microsoft, Chih-Kao Liao of Intertek, Robert Staples of NIST, and Raoul Gabiam, Michael Dimond, Kyle Vitale, Doris Rui, and Matthew Fortes of the MITRE Corporation.

4.2 Proof-of-Concept Server Features

The proof-of-concept server currently implements the following features:

- Two-factor authentication using TOTP and mTLS, which improves the TOTP from ACVP by allowing a user to maintain multiple seeds for simultaneous connections.
- Module registration that defines the security levels, embodiment, and other properties of the cryptographic module, and automatically determines which TEs are applicable to the cryptographic module.
- Module evidence submission that prompts a client to provide evidence addressing TEs that are applicable to the cryptographic module, and will show which TEs have not yet been addressed by the submission to ensure completeness.
- Module security policy submission defined entirely in JSON, which will generate the security policy automatically, allowing the client to retrieve the completed document, and ensuring that all sections are present and completed.
- Automatic processing of functional test evidence (FT-TEs) based on the test type selected by the lab.
- Accepts source code test evidence (SC-TE) based on the test procedure selected by the lab.
- Provides endpoints that list the accepted schemas for submission endpoints and defines a query to obtain a specific version of a schema.
- Handles other documentation test evidence (OD-TE), rounding out all the evidence types defined by the TE Workstream.
- Applies an automated rule checking engine on completed submissions. This is introduced on the ACMVP Demo Server as a proof of concept to do things like ensure a submission is consistent with itself. Submissions for a full module can be very large, needing several hundred TEs addressed, cryptographic algorithm and entropy source validations, etc. The rule checks can be expanded to complete cross-references to algorithm and entropy certificates to ensure that all content in the current request is accurate. Much of this work is done manually by a reviewer

looking at the publicly available algorithm validation certificate and the registered capabilities of the submitted cryptographic module.

4.3 Server Implementation

The server uses much of the same infrastructure as ACVP and ESV, which is intended to keep the same team available to maintain the systems once they are integrated by the CMVP. The system is comprised of C# and Python applications along with SQL Server databases.

The server implementation can be broken down into two major applications. The first is WebPublic, the front-facing application that serves the application programming interface (API) with which clients interact. This application handles HTTPS requests from users to retrieve or submit data to the CMVP. The second application is the MessageQueueProcessor. As tasks to create or update data are collected through WebPublic, they enter a queue. The MessageQueueProcessor reads those requests using a first-in, first-out ordering to fulfill them. This handles the core logic of creating modules, applying the TE Filter, building security policy documents, and running the core automation checks on a module seeking validation.

4.4 Client Implementations

This section describes the two open-source clients, Libamvp and ACVP Proxy, that provide foundational code for developers to build upon when interfacing with the server.

4.4.1 Libamvp – Cisco

Libamvp is an example client for the AMVP protocol developed by Cisco engineers. It is C-based and interacts with the server by parsing user-generated JSON and is intended to be a simple tool to showcase the protocol and assist developers as they create workflows for the generation and submission of AMVP data. Libamvp can create modules and certification requests, submit all required evidence and security policy information, retrieve security policy PDFs, check for the status of a certification request, and perform other actions, as development continues.

Libamvp can be found here: https://github.com/cisco/libamvp.

4.4.2 ACVP Proxy – atsec information security

The client is called the ACVP Proxy and is supported by atsec information security. The name is ACVP Proxy because this is a continuation of an older project designed to interact with the NIST ACVP servers. It now provides the interface to access the NIST ACVP, ESV, and ACMVP services. The code is open source and available at the public repository: https://github.com/smuellerDD/acvpproxy.

The ACVP Proxy has many options, allows a flexible deployment, and is extendable to cover an arbitrary number of IUT definitions. The ACVP Proxy implements the entire interaction with the NIST servers to obtain the data from the server and upload all required data to the server.

4.5 Accessing the ACMVP Demo Server

Detailed instructions on accessing the ACMVP Demo Server hosted by NIST can be found at https://pages.nist.gov/ACMVPDocs/protocol/index.html#accessing-the-acmvp-demo-server.

5 Research Infrastructure

The infrastructure workstream team adopted an iterative approach to modernize the CMVP supporting infrastructure. Each iteration introduced progressively advanced architectures, leveraging cloud-native services to improve scalability, portability, deployment speed, and security, all while ensuring cost efficiency. The modernization efforts have resulted in a containerized application that has been successfully deployed on the Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) platforms. The final iteration features Amazon EKS leveraging Bottlerocket Amazon Machine Images (AMIs) with FIPS 140-3 compliance enabled.

Furthermore, the modernized architecture integrates a managed database service to enhance operational efficiency and features a fully automated CI/CD pipeline to simplify and streamline server deployments on a Linux platform. Authentication mechanisms have been modernized to incorporate cloud-native solutions, including the AWS Network Load Balancer (NLB).

5.1 Research Infrastructure Workstream Collaborators

The Research Infrastructure Workstream is led by Raoul Gabiam of The MITRE Corporation and Douglas Boldt of Amazon, with contributions from Courtney Maatta, Annie Cimack, Diana Brooks, Charlotte Fondren, Zhuo-Wei Lee, Keonna Parrish, Abhishek Isireddy, Abi Adenuga, Bradley Wyman, Brittany Robinson, Gina McFarland, Damian Zell, Cavan Slaughter, Rayette Toles-Abdullah, Keith Hodo, John Dwyer, Ahmed Virani, Daftari Mrunal, Kasireddi Srikar Reddy, Srujana Alajangi, and Natti Swaminathan of Amazon; Robert Staples and Murugiah Souppaya of NIST; Jason Arnold of HII; Michael Dimond, Kyle Vitale, Phillip Millwee, and Josh Klosterman of the MITRE Corporation; and John Booton, Aaron Cook, and Jeffrey LaClair of ITC Federal.

5.2 Modernization Approach

The existing CMVP production environment was initially deployed in a data center internal to NIST. A subset of the environment that was providing services to the test labs was virtualized and migrated to AWS GovCloud to take advantage of the high availability and resiliency offered by cloud infrastructure. The CMVP system administrators have maintained the AWS infrastructure for several years.

The modernization journey started with a complete inventory and understanding of the existing production environment in AWS, including all the virtualized assets, the network, data flows, functionalities, and dependencies. Once the existing architecture was fully documented, it was replicated in a research environment managed by the NCCoE team to establish an initial baseline that could be analyzed, and opportunities were identified to incrementally modernize the application and supporting infrastructure throughout the lifecycle of this project. The NCCoE research is performed in AWS to ensure the findings can be easily replicated in the production environment. The objective is to deliver the new capabilities required at the application level to support the Protocol Workstream while maintaining some compatibility with the existing production environment.

5.3 Replication of the Legacy Production CMVP Environment

This section gives historical context to the ACMVP application. The production CMVP AWS environment was replicated to the current ACMVP research environment, which set a baseline from which modernization opportunities were identified.

Figure 2 represents the baseline architecture present in the research environment before modernization efforts.

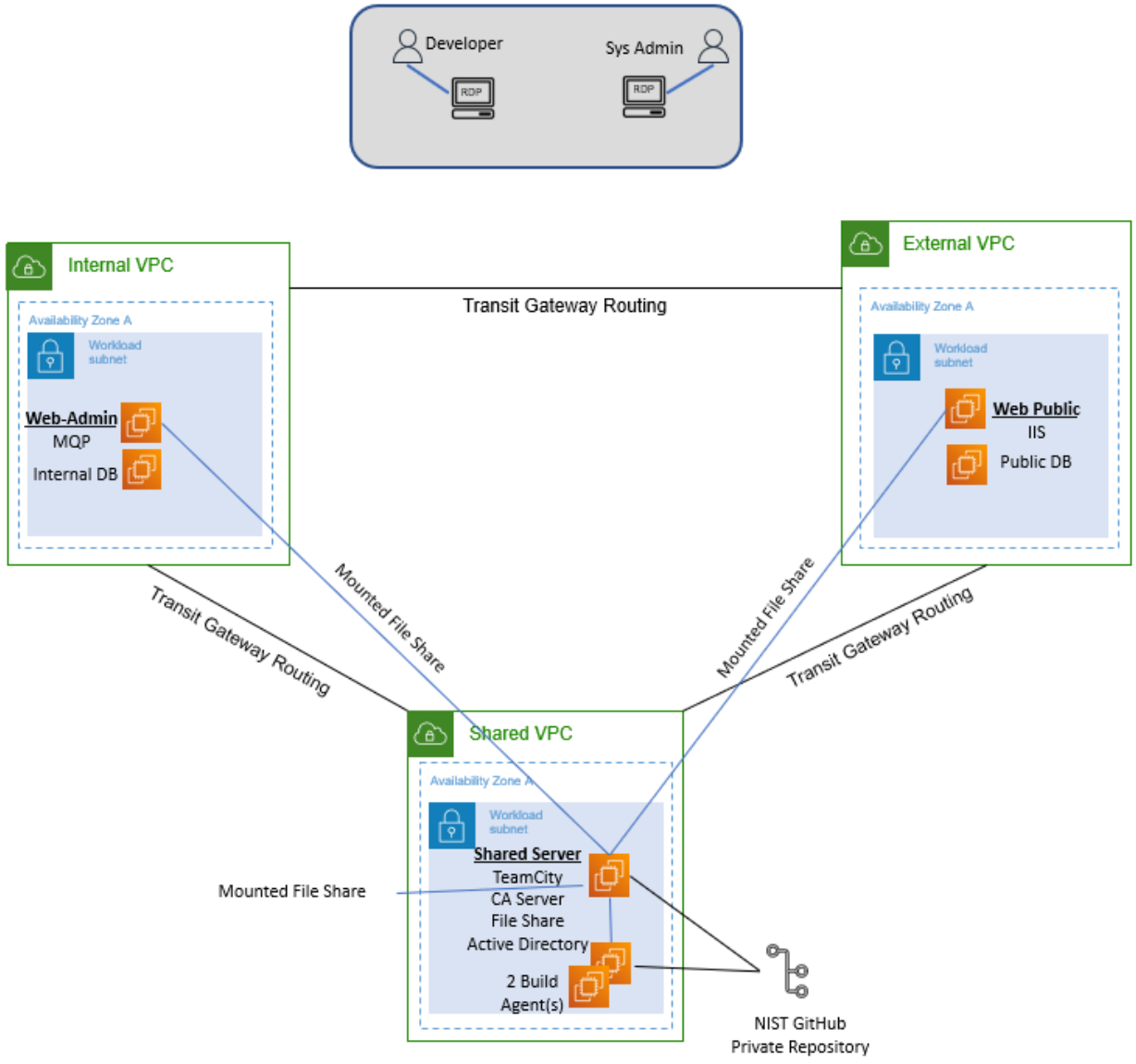


Figure 2 Legacy System Architecture Diagram

The External Amazon Virtual Private Cloud (VPC) handles any public-facing applications and utilities, including the WebPublic application (sitting underneath Microsoft IIS) and the public database. These services are split into two separate Amazon EC2 instances.

The Internal Amazon VPC hosts private applications and utilities, including the MessageQueueProcessor (MQP) application and the internal database. These services are split into two separate Amazon EC2 instances.

The Shared Amazon VPC hosts shared applications and utilities, including JetBrains TeamCity for CI/CD, the Certificate Authority (CA) server, the file share service for backups and logs, and the Microsoft Active Directory service, which is hosted on one Amazon EC2 instance in the research environment for the sake of simplicity.

Figure 3 details the steps in the workflow that occur when the user submits a request, which are listed in this document to describe the necessary tools and their use cases in the critical workflow.

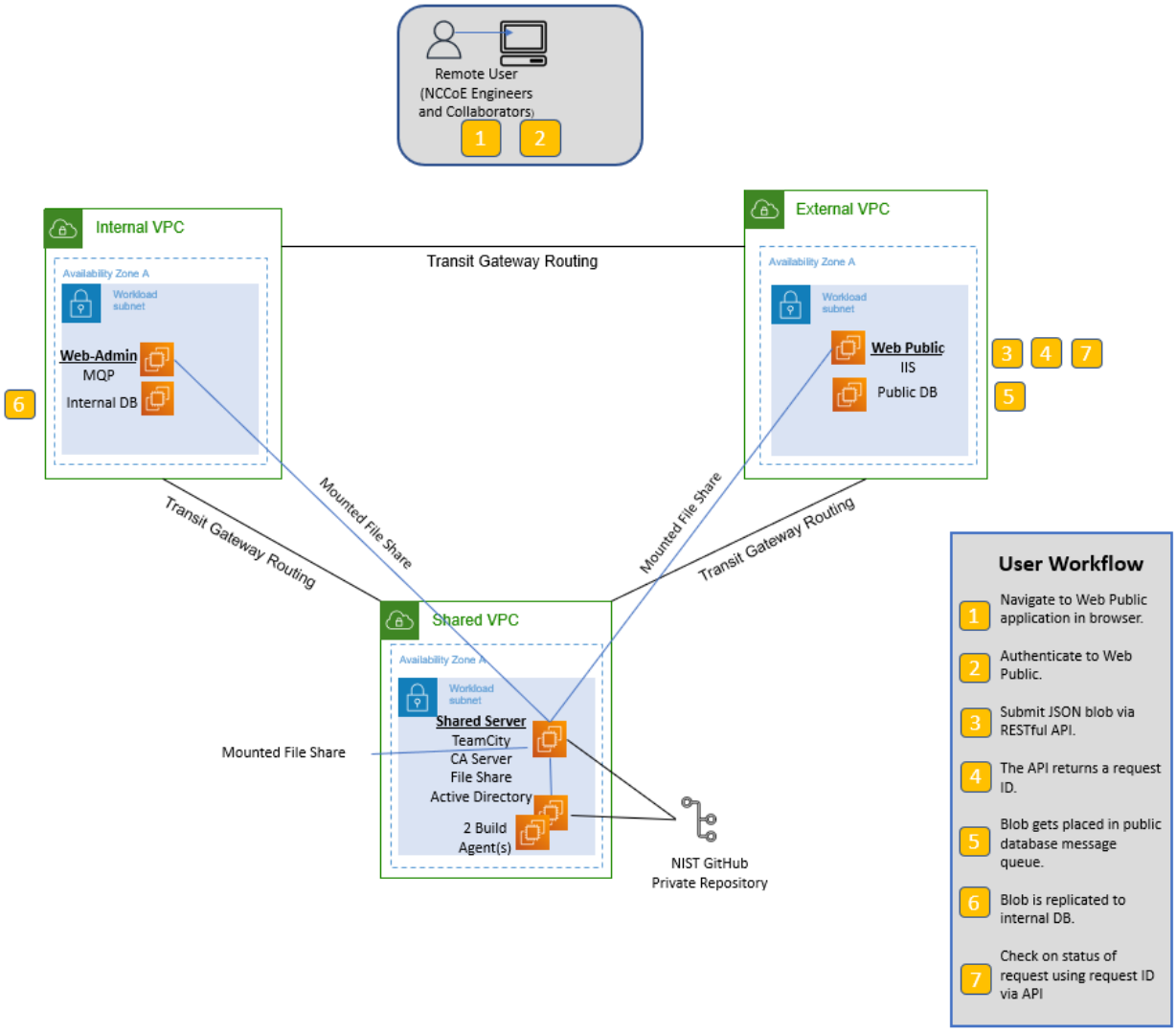


Figure 3 Legacy System End User Workflow

WebPublic is publicly available for registered NVLAP users to submit their requests, which will include authentication requests that are partially handled by Microsoft IIS for Windows Server through mutual TLS (mTLS). Microsoft IIS receives its server-hosting certificate through the CA Server. The application

stores and retrieves data from the Public DB as needed by the requests it receives. Any stored data is replicated to the Internal DB through the encrypted message queue (MQ). The MQP processes the request and stores necessary changes to the Internal DB, which is replicated to the Public DB for user retrieval. Logging occurs throughout the process, tracking the request and where the processing is in the WebPublic or MQP application. These logs are stored on a file share for access by a system administrator, along with database backups.

5.4 AWS Target Architecture by Service

This section maps services in the baseline legacy infrastructure to equivalent services provided by AWS. Due to the CMVP system administrators' familiarity with hosting environments in AWS, the research was focused on AWS-based solutions. While this document only addresses AWS services, equivalent services could be found in other cloud providers.

Table 11 provides the mapping between services used in the legacy ACMVP research environment and equivalent services offered by AWS. A more detailed explanation of the mappings can be found below. Explanations are provided for selected mapped services. Services in bold were modernized to equivalent versions, and services in italics were not selected for modernization.

Table 11 Modernized Service Mapping

Required Functionality	Service In Legacy ACMVP	AWS Equivalent Service(s) Considered	AWS Selected Service(s)
Database Service	**Microsoft SQL Server Database**	**Amazon Relational Database Service (RDS) for SQL Server, Amazon Aurora, PostgreSQL**	**Amazon RDS for SQL Server**
Database replication	**Microsoft SQL Server Replication**	**AWS Database Migration Service (DMS)**	**AWS DMS**
Code build/deployment	**JetBrains TeamCity**	**AWS CodePipeline, AWS CodeBuild**	**AWS CodePipeline and AWS CodeBuild**
Public Facing application	**WebPublic API server**	**Containerized Application, Amazon Elastic Container Service (ECS), Amazon Elastic Kubernetes Service (EKS), Amazon Lambda**	**Amazon ECS and Amazon EKS**
Internal application	**MessageQueueProcessor Internal server**	**Containerized Application, Amazon ECS, Amazon EKS, Amazon Lambda, Amazon SQS, Amazon MQ**	**Amazon ECS**
Web proxy/ authentication	**Microsoft IIS**	**AWS Application Load Balancer (ALB), AWS Network Load Balancer (NLB), Amazon API Gateway, nginx Reverse Proxy**	**AWS NLB**
Identity Management	*Microsoft Active Directory*	*AWS Managed Microsoft AD*	*Out of Scope*
Domain Name Services	*Microsoft Windows AD DS*	*AWS Route 53 with AWS Managed Microsoft AD*	*Out of Scope.*
File Sharing Services	*Windows File Share*	*Amazon FXs for Windows, Amazon S3, AWS Storage Gateway*	*Out of Scope.* *Note: S3 in use with other AWS*

			services (RDS, Code Pipeline) but not specifically as an alternative to file shares
Code Repository	*Git Repository*	*AWS Code Commit (deprecated), AWS Code Connections*	*Out of Scope.* *Note: AWS Code Connections is being used in conjunction with the existing Git Repository and is not replacing it.*

Equivalent AWS services for the Microsoft SQL Server Database are Amazon RDS for SQL Server, Amazon Aurora, and PostgreSQL. Amazon Aurora only supports MySQL and PostgreSQL, requiring a change from the ACMVP's use of Microsoft SQL Server. Amazon RDS supports a managed version of Microsoft SQL Server. Amazon RDS was selected as the modernization approach due to the existing CMVP code that relies on Microsoft SQL Server.

AWS DMS was selected following the decision to use Amazon RDS to meet the need for data replication. Data replication in Amazon RDS requires AWS DMS, as the instances hosting the databases are managed by AWS and may change IP addresses over time. AWS manages this by providing DNS names to resolve the IP addresses for the databases.

JetBrains TeamCity's equivalent service is mapped to AWS CodeBuild, which was used to provide insight to the CMVP on alternative technologies.

WebPublic had the potential to be containerized or moved to an Amazon Lambda function. The containerized option was selected as it enables local testing, integrates with GitHub, and allows for portability of the codebase. Note that streamlining the deployment process and improving code portability were desired outcomes of the production CMVP infrastructure support team. WebPublic was deployed via a Docker daemon on a NIST Secure Amazon EC2 instance to meet security requirements for a demo server, but Amazon ECS and Amazon EKS were examined as modernization approaches in the research environment. Amazon EKS was selected as the containerization platform offering increased portability and deployment ease.

The MQP was mapped to other MQ services. However, the developed MQP performs functions unique to the ACMVP application, resulting in a decision to containerize the application. AWS Fargate was selected as a technology to convert MQP capabilities to a standalone, containerized service and provide consolidated connection and deployment management.

Microsoft IIS was mapped to AWS NLB, AWS ALB, Amazon API Gateway, and nginx Reverse Proxy. The AWS NLB handles layer-3-requested routing to the application, requiring Microsoft IIS or nginx to process mTLS authentication, or Amazon API Gateway to process API keys as an alternative mode of authentication. The AWS ALB was examined as a way to handle both mTLS authentication and the routing to the containerized WebPublic application. The AWS ALB and other tools may still meet the requirements, but were not explored further.

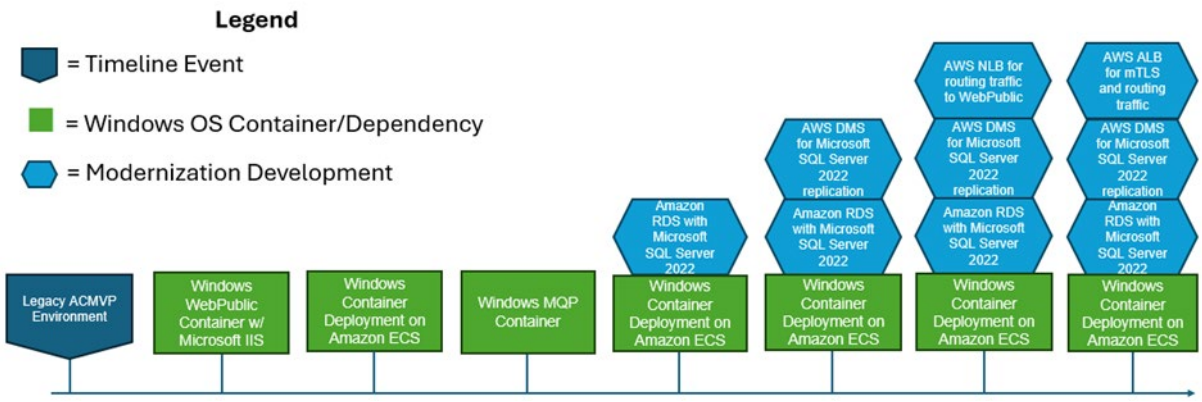
While equivalent services were identified for GitHub, Microsoft Active Directory, Microsoft Windows AD DS, and File Share, these services were determined out of scope as they were already well established within the environment.

5.5 Key Modernization Components

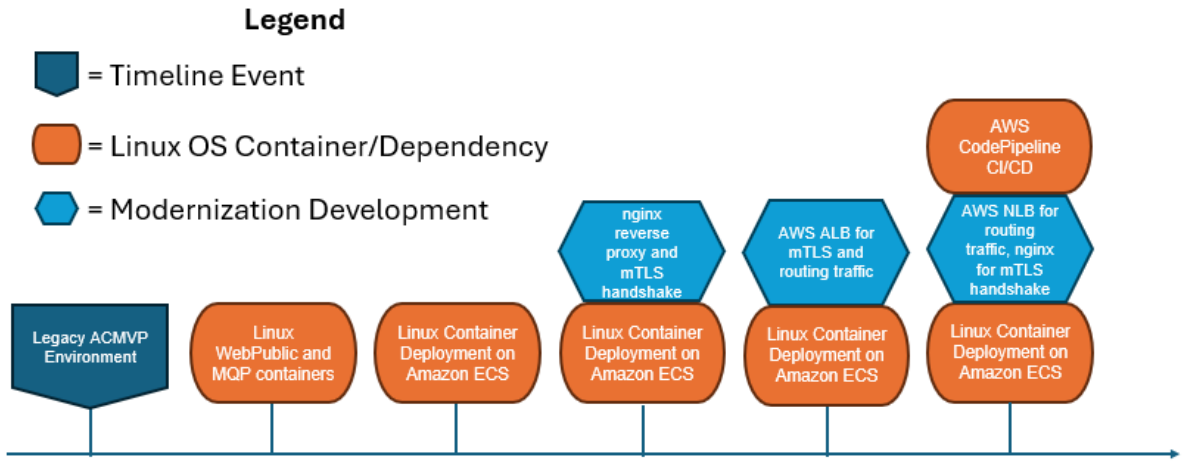
This section describes the modernization research journey of the ACMVP application, which is presented in three progressive iterations. As the application is a REST API with a backend database and MQP, similarly structured applications can utilize this research in making informed decisions to update, improve, or otherwise modernize their infrastructure.

In the first iteration, the application was containerized on Amazon ECS in a Windows container as a proof of concept. In the second iteration, the application was further decoupled and containerized as a Linux application on Amazon ECS to enable a CI/CD integration, which required Docker in Docker (the practice of running a Docker engine inside a Docker container) and is only supported on Linux. In the third and final iteration, the containerized Linux application was deployed on Amazon EKS, tailored for NIST production. Furthermore, the supporting infrastructure and application deployment were bundled as Infrastructure as Code (IaC) to streamline and automate deployment. Lastly, the final iteration features Amazon EKS with Bottlerocket Amazon Machine Images (AMIs), ensuring a FIPS 140-3 compliant environment.

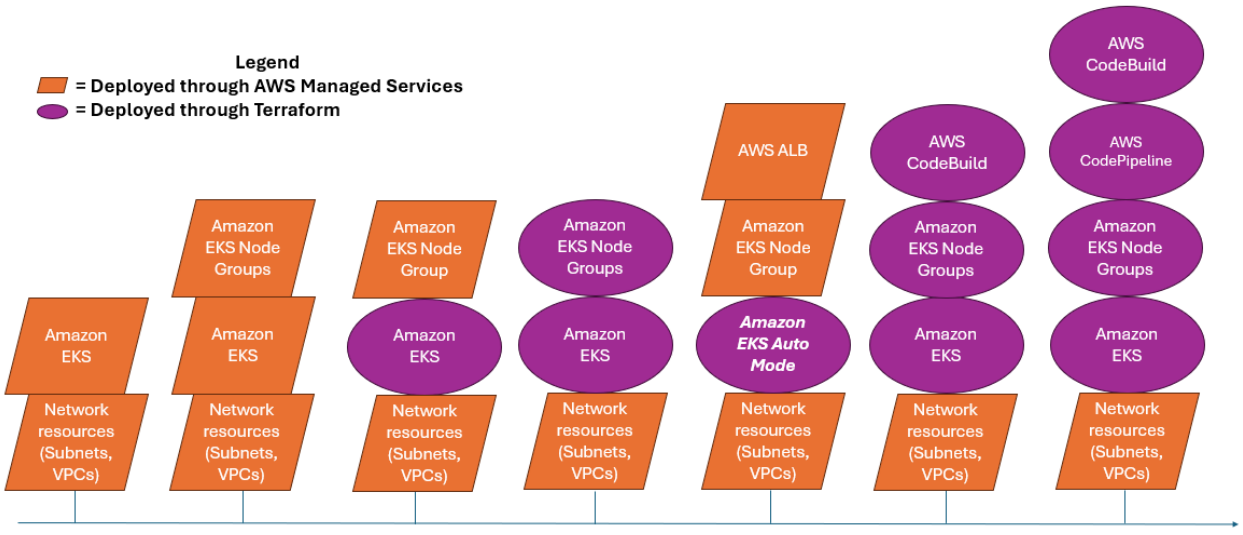
Figures 4, 5, and 6 depict the three progressive modernization iterations. A pentagon flag in dark blue represents a timeline event, a green rectangle represents a Windows OS container development, a cyan hexagon represents a general modernization development, and an orange ellipse represents a Linux OS container development. Note that AWS CodeBuild and AWS CodePipeline CI/CD are in orange, as they only apply to Linux OS containers, as explained within the Application Deployment Modernization section. Purple ovals describe resources deployed by Terraform and IaC automation. Orange parallelograms represent deployments through AWS-managed services.



1200 **Figure 4 First iteration: Windows Container OS Modernization Progression on ECS**



1201 **Figure 5 Second iteration: Linux Container OS Modernization Progression on ECS**



1202 **Figure 6 Third and final iteration: Deployment on EKS Auto Mode with FIPS 140-3 compliance enabled**
1203 **Progression of Experiments**

Figure 7 depicts the AWS services and tools used in the modernized system architecture. The CMVP services are appropriately divided into separate Amazon VPCs. The External Amazon VPC provides the outward-facing service layer (i.e., WebPublic and nginx), which is hosted on AWS EKS. External connectivity is enabled by an Amazon Elastic Load Balancer (ELB) instance. Additional infrastructure services consist of CI/CD automation provided by AWS CodeBuild, a single EC2 instance for management functions, and an external instance of Amazon RDS.

The Internal VPC provides RDS replication via AWS Database Migration Services (DMS) and the internally-hosted Amazon RDS instance. An EC2 instance provides infrastructure management for internal resources. Finally, the CMVP MQP service is hosted on Amazon ECS.

Figure 7 also depicts the AWS cloud-managed services that CMVP relies on for deployment and management of internal and external resources. Database backups, CI/CD artifacts, and data are hosted in Amazon S3. AWS CodePipeline and CodeBuild provide the CI/CD automation for internal and external resources. AWS Elastic Container Registry provides scalable, fault-tolerant storage for container images that provide WebPublic, MQP, and nginx functionality to the project.

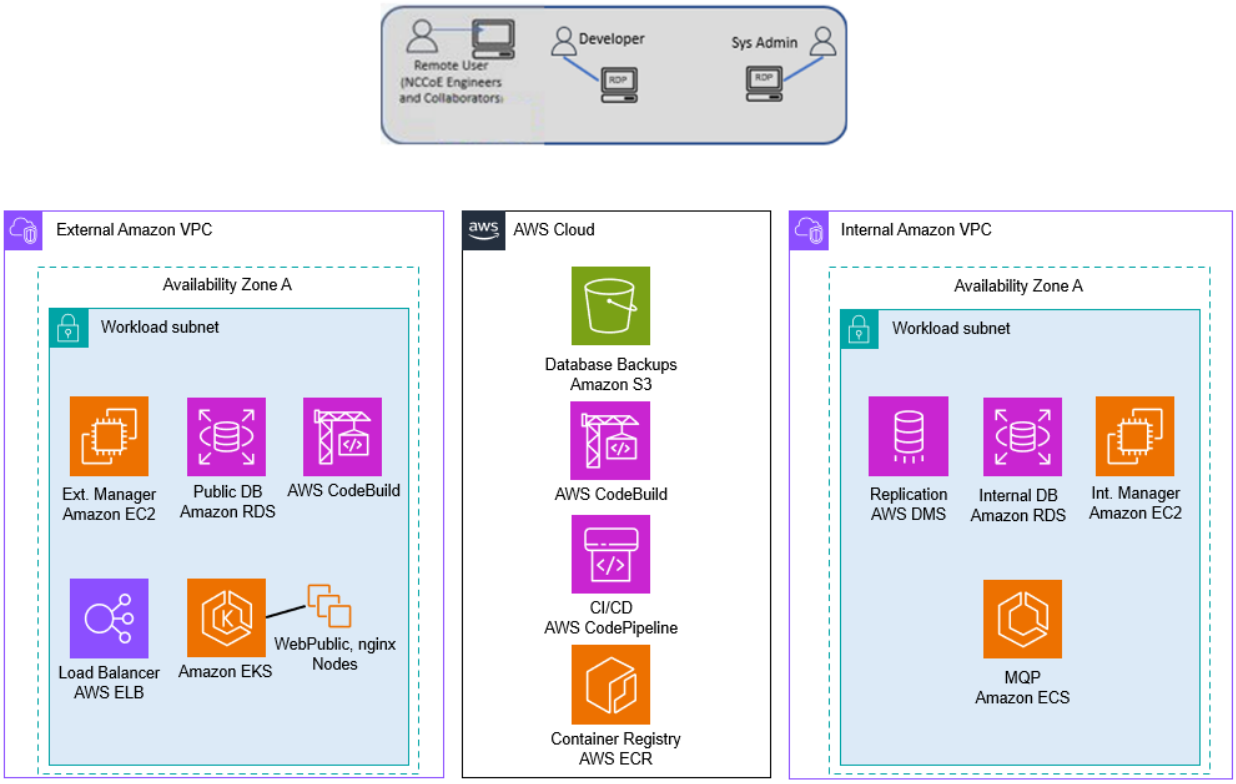


Figure 7 Modernized System Architecture

Figure 8 depicts the desired client workflow through the modernized resources. The client connects to an AWS Elastic Load Balancer (ELB), whose destination is open to the public. AWS ELBs refer to both ALBs and NLBs, depending on the use case. The load balancer forwards the traffic to the WebPublic application, running through one of the launch types identified in the *Application Deployment Modernization* section. This application uses its connection to the Public Database to store the data passed through by the client. AWS DMS, residing in the Internal Amazon VPC, replicates that

information to the Internal Database through the MessageQueue table. The MQP recognizes the new items in the queue and processes them, finishing its processing by storing updates back into the Internal Database. These updates are replicated back into the External Database through the AWS DMS instance. Once updates are populated into the External Database, clients can view those changes through their original connection workflow.

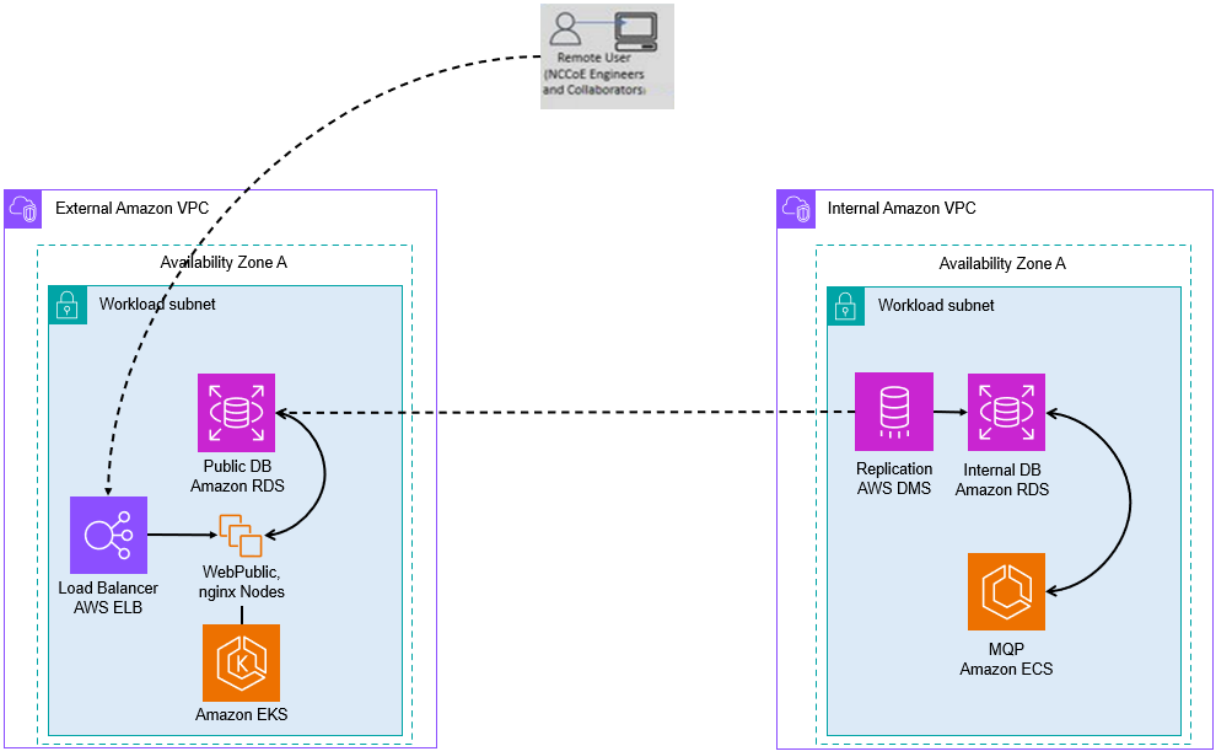
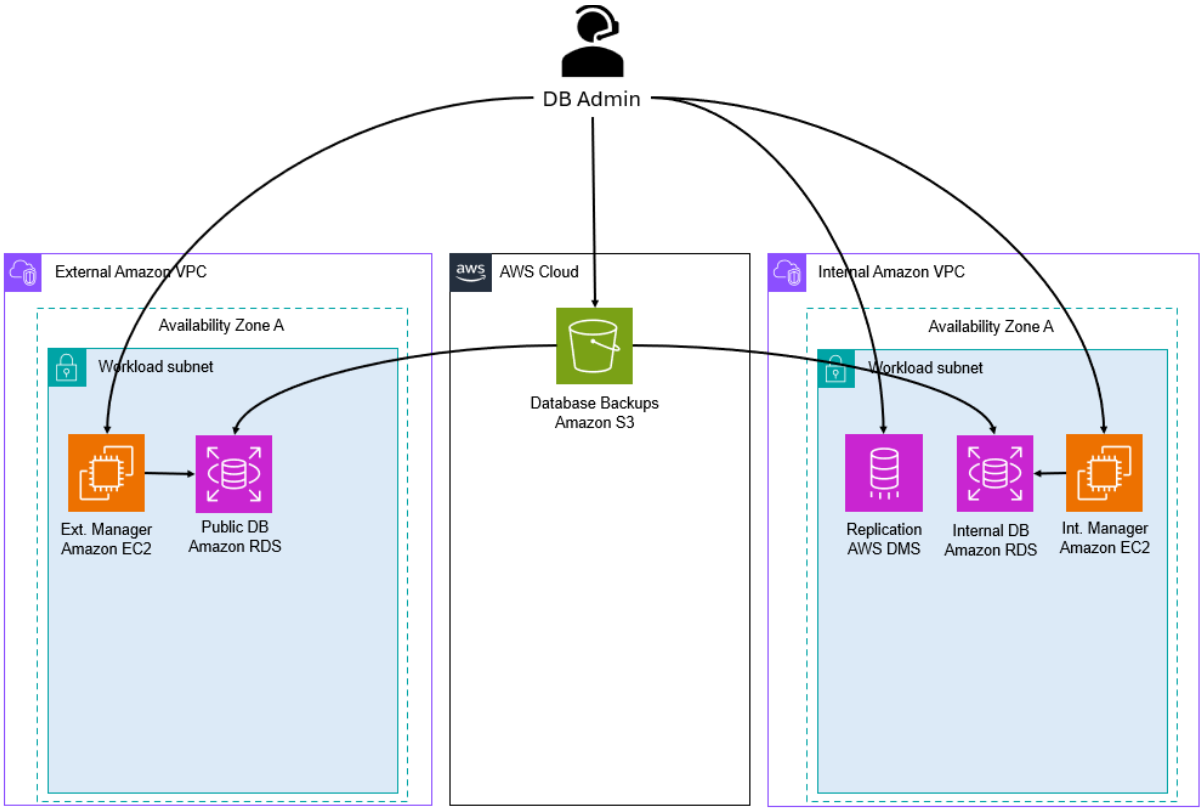


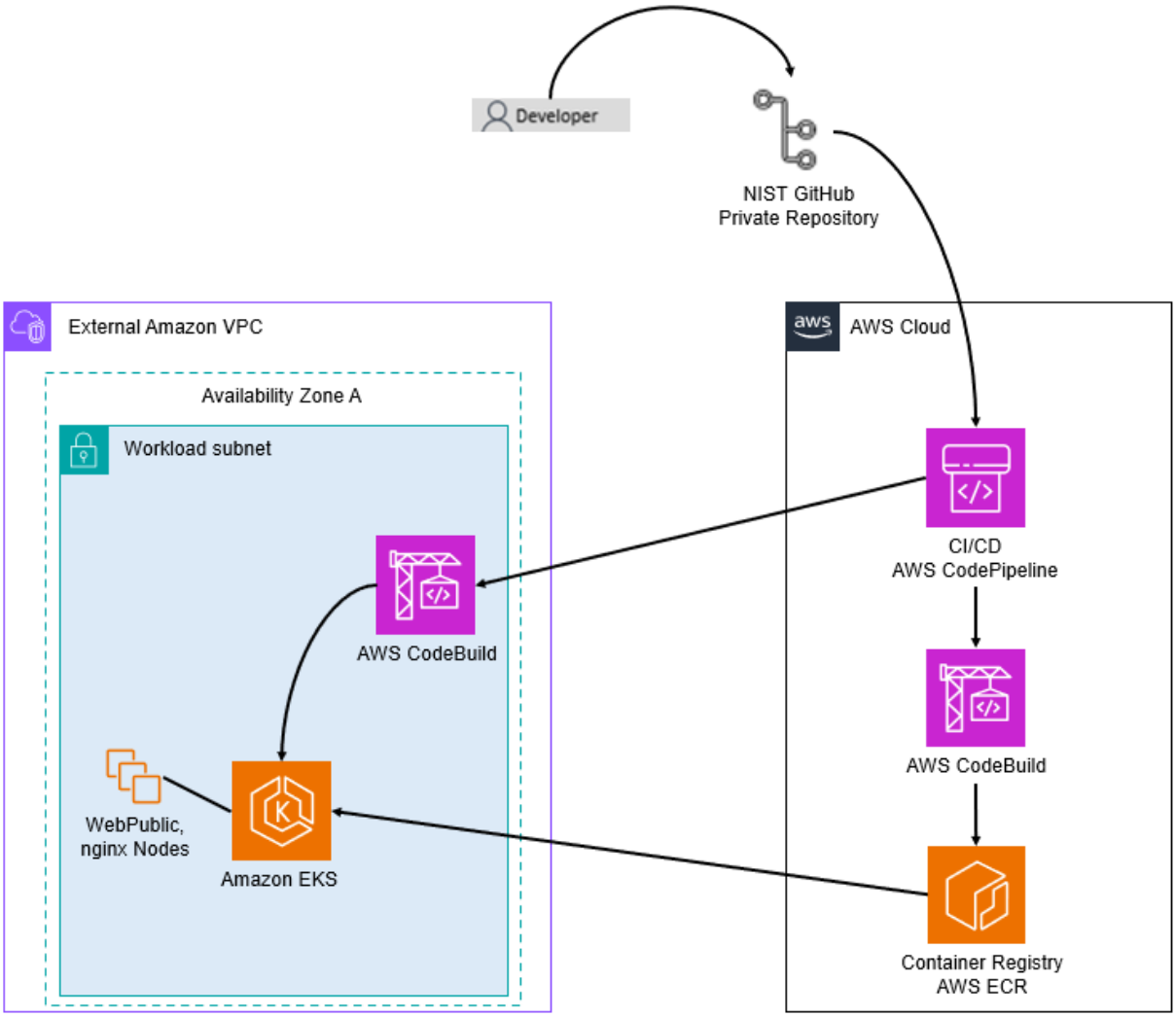
Figure 8 Modernized Client Workflow

Figures 9 and 10 depict the different workflows the system administrator and the developer take to implement updates to the application code or database.



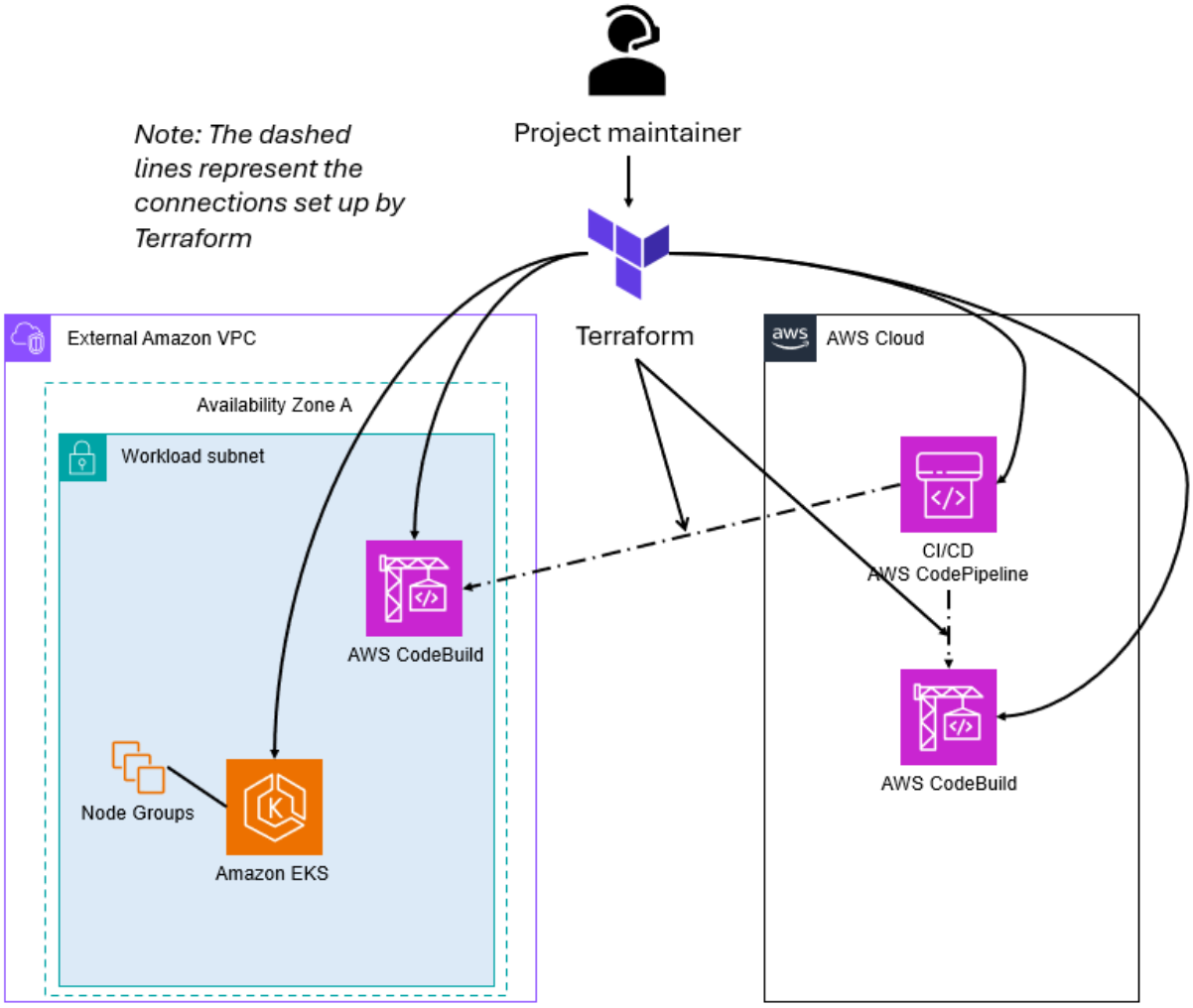
1233 **Figure 9 Modernized Database Administrator Workflow**

1234 To make database changes, a developer would generate a backup of the database they would like to
1235 deploy in the modernized environment. This backup would be given to the system administrator, who
1236 would place the backup into a private Amazon S3 bucket. The system administrator can then connect to
1237 a database connector, where the backup can be retrieved from Amazon S3 and deployed into the
1238 Amazon RDS instance. This process requires AWS DMS replication to be reinitiated for the new set of
1239 desired tables.



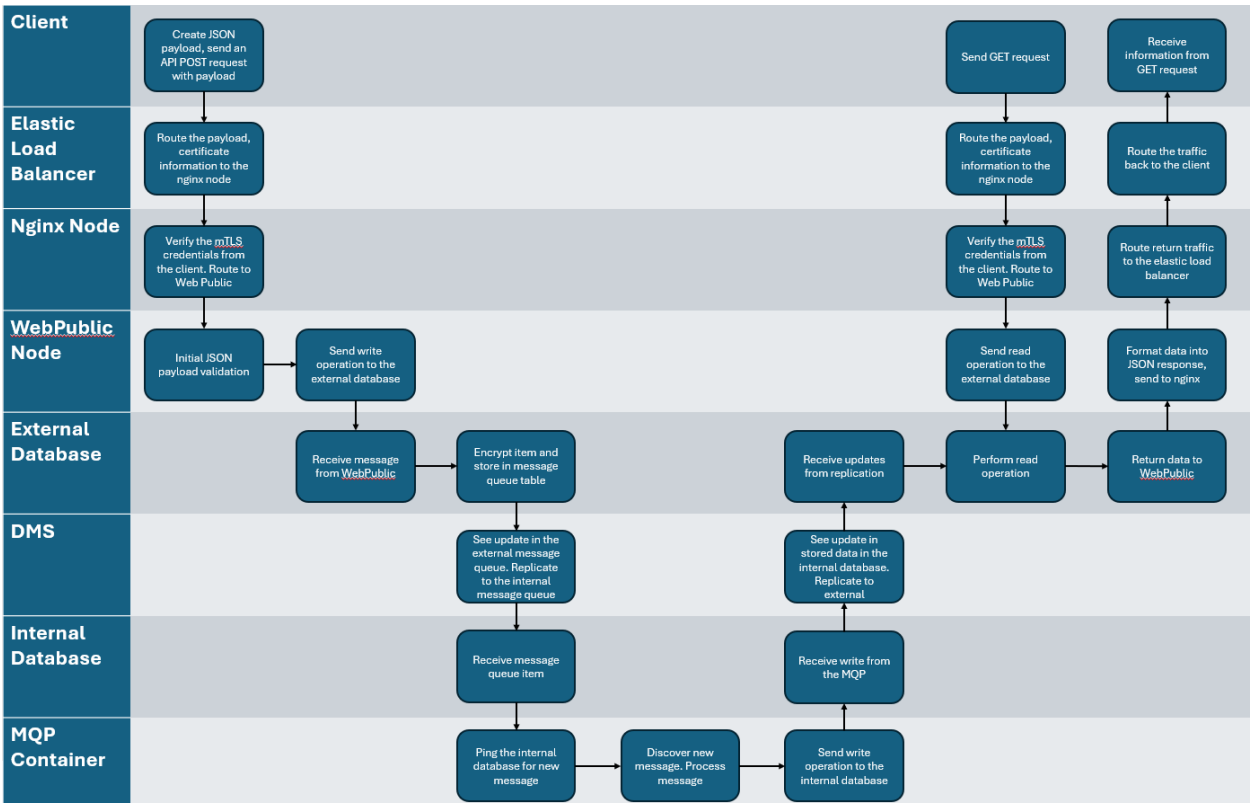
1240 **Figure 10 Modernized Developer Workflow**

1241 To make code changes, a developer would push their changes to a code repository, like GitHub. From
1242 there, a container build is completed either locally by a system administrator or through the AWS
1243 CodeBuild/CodePipeline, where a container image is created and stored in the Amazon Elastic Container
1244 Registry (ECR). Once those changes are pushed, new tasks can be started (manually or automatically)
1245 with the updated application code.



1246 **Figure 11 Modernized Project Maintainer Workflow**

1247 Figure 12 below summarizes the modernized workflows in a swim lane diagram.



1248 **Figure 12 Modernized Architecture Swim Lane Diagram**

1249 5.6 CI/CD Pipeline Modernization

1250 AWS CodeBuild and AWS CodePipeline are used to automate the continuous integration and
1251 deployment (CI/CD) process. The AWS CodePipeline creates a workflow that's structured into multiple
1252 stages and ensures code tracking, containerized builds, artifact storage, and automated deployment of
1253 CMVP services.

1254 **Source Control & Change Detection – GitHub + AWS CodePipeline:** AWS CodePipeline is integrated with
1255 GitHub, allowing it to automatically detect new code changes in the repository. When a developer
1256 pushes new code, AWS CodePipeline triggers the pipeline execution, automated build actions via AWS
1257 CodeBuild, automated deploy actions to Amazon EKS, or automated deploy actions to AWS CodeDeploy.

1258 **Build & Containerization – AWS CodeBuild + Amazon ECR:** AWS CodeBuild is used to build Docker
1259 containers based on the latest code changes. The build process includes compiling, testing, and
1260 packaging the application into containerized images. These images are then tagged and stored securely
1261 in Amazon ECR for deployment.

1262 **AWS ECS Deployment & Orchestration – AWS CodeDeploy + Amazon ECS:** AWS CodeDeploy handles
1263 the deployment of containerized applications into Amazon ECS. Amazon ECS ensures that the latest
1264 container versions are automatically deployed and scaled across available compute resources.

AWS EKS Deployment & Orchestration – AWS CodeBuild + Amazon EKS: AWS CodeBuild handles the automated deployment of Kubernetes services to AWS EKS. As with the Amazon ECS example, AWS EKS ensures that the latest container versions are automatically deployed to available compute resources.

5.7 Database Modernization

Database modernization focuses on modernizing the hosting environment for the database service. The application requires an internal and external database with replication of data between the two to communicate updated information.

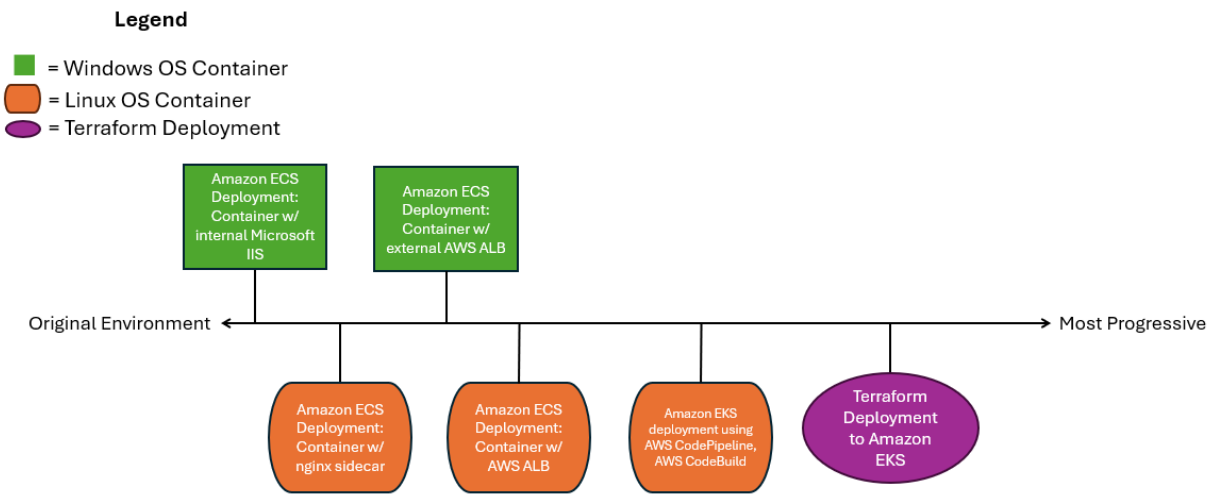
Amazon Relational Database Service (Amazon RDS): The Microsoft SQL Server 2019 edition in the ACMVP demo environment has been replaced with Amazon RDS for SQL Server 2022, with a standard license.

AWS Database Migration Service (AWS DMS): Microsoft SQL Server allows for native data replication in the legacy ACMVP research environment. However, the migration to Amazon RDS necessitates a new data replication service because the underlying resource hosting the database is not owned by the customer, but by AWS. AWS DMS maintains replication between the Amazon RDS databases.

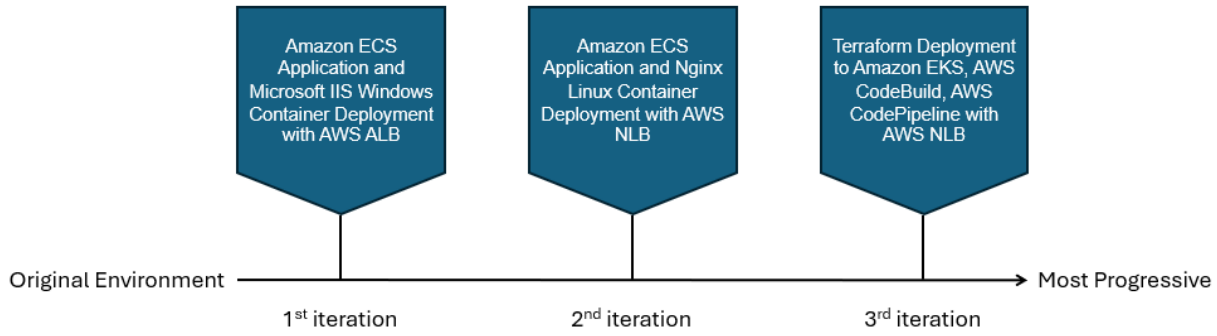
5.8 Application Deployment Modernization

The application deployment modernization focuses on containerizing the WebPublic and MQP applications. Utilizing containers provides benefits and options such as blue/green deployments, vulnerability scanning the images in a registry in advance of deployments, and shorter exposure times from routine deployments.

Figure 13 demonstrates the progression of the approaches taken to modernize the application into a container. The markers on the top represent the Microsoft Windows Container, while the markers on the bottom represent the Linux Container.



1287 **Figure 13 Application Modernization Progression**



1288 **Figure 14 Progression of Containerization Builds**

1289 The first iteration is the closest to the original ACMVP environment. It is a Microsoft Windows container
1290 on Amazon ECS that encapsulates both the application and the Microsoft IIS proxy to authenticate and
1291 route traffic. This solution containerizes the precise environment that exists in the WebPublic Amazon
1292 EC2 instance.

1293 The AWS ALB lifts the proxy services into the cloud while performing the mTLS handshake. Client
1294 certificate details are passed at the application layer as headers to the application, which are then used
1295 for performing authentication for the application.

1296 The second iteration is a Linux container on Amazon ECS with an nginx sidecar container. It advances the
1297 environment by offering a smaller container image size and proxy being utilized, and allows for the
1298 container or nginx to be modified without causing the other to be taken offline, decoupling the
1299 application.

1300 The AWS NLB lifts the proxy services into cloud services. This approach allows AWS NLB to pass TLS
1301 traffic through to the containerized nginx/WebPublic to handle the mTLS handshake.

The third and final iteration applies the progress made in the first two to a NIST production-ready architecture. Kubernetes being the preferred containerization platform at NIST for production systems, the containerized CMVP application is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Furthermore, Amazon EKS is used with Bottlerocket AMIs to ensure a FIPS 140-3 compliant environment.

Details regarding the services explored can be found in Appendix D. Application Modernization.

5.9 Microservice Architecture

The architecture's final iteration marked a transition to the broad use of containers and microservice architecture solutions. Both software development and infrastructure management can account for the use of similar containerized services running lightweight versions of WebPublic, MQP, and nginx applications.

The software development support for this architecture can leverage locally hosted Kubernetes installations (i.e., Docker Desktop or Rancher Desktop). Both Docker Desktop and Rancher Desktop are commercial and open-source alternatives used for the development of containerized applications and microservices. Both solutions provide a localized Kubernetes cluster, APIs, compute, and storage resources. These solutions provide the software developers with similar provisioning and deployment patterns as well as APIs that provide an identical management plane when compared to production instances of the application.

Amazon EKS provides the cloud-hosted microservice architecture used in the final iteration of CMVP infrastructure modernization. Amazon EKS runs Kubernetes under the hood. The access patterns and APIs all behave in a similar fashion to other Kubernetes implementations, such as Docker Desktop or Rancher Desktop. Amazon EKS also provides access and configuration controls that can be managed through the Amazon Console or with Infrastructure as Code (IaC).

5.10 Infrastructure as Code (IaC)

Automation and management of AWS resources, services, and infrastructure was a focus of the components referenced throughout Section 5. Infrastructure as Code (IaC) was selected to handle the standardization and automatic configuration of both AWS and CMVP components. Terraform was chosen as the IaC declarative configuration language (DCL) to represent these resources and components.

Terraform modules were used to decompose the CMVP application into groups of common resources. At the foundation, there is a first (e.g., root) module that dictates what groups of Terraform modules and top-level resources will represent a single deployment of the project's infrastructure. Each module, when called, will deploy and configure resources that are tightly associated with one another. The following list describes the components that are called by the root module:

EKS Module: This consists of Amazon EKS cluster services and dependencies. This module also deploys Amazon EKS Node Groups, which serve as compute (i.e., Amazon EC2) instances that support applications deployed to an EKS cluster. Dependencies include security group and AWS ELB target group attachments that connect the EKS cluster to networks and load balancer capabilities.

CodeBuild Docker Module: This is the IaC that's shared with the CodeBuild Deploy Module that is referenced later in this section. This module configures AWS CodeBuild services and dependencies. This includes attachments to existing AWS Identity and Access Management (IAM) roles that provide permissions to AWS CodeBuild and other AWS services and resources. This module also includes the build specification (i.e., buildspec) that scripts the actions that AWS CodeBuild performs as part of the Continuous Integration/Continuous Delivery (CI/CD) process.

CodeBuild Deploy Module: As previously stated, this module's code is shared with the CodeBuild Docker Module. This module configures AWS CodeBuild projects and dependencies. Similar to CodeBuild Docker, this includes configuration of permissions via IAM role attachments as well as CI/CD build specifications. This CodeBuild module is configured to attach to the existing Amazon VPC where the Amazon EKS cluster and dependencies are deployed. This enables network access to Amazon VPC resources to support containerized deployments.

CodePipeline Module: This module handles the CI/CD orchestration of the AWS CodeBuild projects previously mentioned. AWS IAM roles are assigned to a single pipeline to permit access to the AWS services and resources that are required for CI/CD automation. CodePipeline is also configured to access specific source code branches of specific instances of the application (i.e., dev, test, or production) or isolated deployments used for testing purposes.

6 Findings and Recommendations for Future Work

6.1 TE Workstream outputs

The Test Evidence Workstream demonstrated how defining a structured approach to Test Evidence categorization based on security level and module type enables filtering of required Test Evidence for validation. This targeted approach enhances validation efficiency while maintaining rigorous security standards.

Testing procedures and classifications of TEs reduce reviewer overhead by allowing them to focus only on requirements that cannot be easily automated. With the addition of the TE Filter, reviewers will not need to decide if requirements are applicable because those decisions have already been made prior to submission. The CMVP is determining how the new assurances collected will affect and shape the future of module reviews.

An interested community such as the CMUF may define standardized test procedures for particular TEs to further streamline automation. Consistent test methodologies enable scripting and automation to provide pre-formatted outputs that conform with the protocol.

6.2 Protocol and Development outputs

The Protocol Workstream demonstrated how an application can process a well-defined and structured payload and provide instantaneous feedback on the completeness of the required data submitted for module validation. The following two key features are being integrated into the production CMVP environment:

The server implementation will act as a new front door for CMVP submissions with built-in checks for completeness and accuracy. Labs have the ability to respond to issues instantly before they become comments from a reviewer.

An excerpt of the implementation of the TE Filter, called the Requirements Library, can be integrated into other projects. A NuGet package has been created from the ACMVP code and served internally as a resource for CMVP developers. The package implements the TE Filter and tracks all FIPS 140-3 requirements.

Future CMVP work includes incorporating other module submission types, such as those that affect existing validations. Additionally, updates to WebCryptik are necessary to help build the evidence payloads required for ACMVP.

6.3 Infrastructure outputs

The Infrastructure Workstream demonstrated an iterative approach to modernizing the CMVP supporting infrastructure, enabling operational efficiency, portability, reproducibility, and CI/CD integration, while adhering to production-ready security compliance requirements. The close collaborative effort with the production CMVP team has resulted in showcasing desired capabilities, jump-starting the production infrastructure modernization. Cloud-native technologies, tools, and concepts demonstrated and covered in this document are being implemented in the production CMVP

environment in collaboration with the NIST Infrastructure and Security teams as a testament to the operational efficiencies a modernized environment offers.

The approach and capabilities developed have an impact beyond CMVP and are applicable across NIST and the industry to modernize existing infrastructure and services.

7 References

[1] National Institute of Standards and Technology, "Federal Information Processing Standards Publications (FIPS PUBS) 140-3: Security Requirements for Cryptographic Modules," 2019. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.140-3.

[2] National Institute of Standards and Technology and Canadian Centre for Cyber Security, "Draft FIPS 140-3- Cryptographic Module Validation Program Management Manual, Version 1.1," July 2022. [Online]. Available: https://csrc.nist.gov/Projects/cryptographic-module-validation-program/cmvp-fips-140-3-management-manual.

[3] ISO, ISO/IEC 24759:2017: Information Technology - Security Techniques - Test Requirements for Cryptographic Modules, Geneva, Switzerland: International Organization for Standardization, 2017.

[4] National Institute of Standards and Technology, "CMVP Documentation Requirements: CMVP Validation Authority Updates to ISO/IEC 24759." NIST Special Publication 800-140A. National Institute of Standards and Technology (NIST). https://doi.org/10.6028/nist.sp.800-140a.

[5] ISO, ISO/IEC 19790:2012: Information Technology - Security Techniques - Security Requirements for Cryptographic Modules, Geneva, Switzerland: International Organization for Standardization, 2012.

[6] National Institute of Standards and Technology, "Cryptographic Module Validation Program (CMVP) Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B." NIST SP 800-140Br1. National Institute of Standards and Technology. https://doi.org/10.6028/nist.sp.800-140br1.

[7] National Institute of Standards and Technology and National Cybersecurity Center of Excellence, "Automation of the Cryptographic Module Validation Program," September 2022. [Online]. Available: https://www.nccoe.nist.gov/automation-nist-cryptographic-module-validation-program.

[8] National Institute of Standards and Technology and Canadian Centre for Cyber Security, "FIPS 140-3 Regression Testing Table to Be Upcoming in FIPS 140-3 Management Manual," NIST, Washington D. C., 2022.

[9] National Institute of Standards and Technology, "CMVP Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B (2nd Public Draft)," October 2022. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-140b/rev-1/draft.

[10] National Institute of Standards and Technology, "CMVP Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B," May 2022. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-140b/rev-1/archive/2022-05-12.

[11] NIST Handbook (HB) 150-17, NVLAP Cryptographic and Security Testing, https://doi.org/10.6028/NIST.HB.150-17-2020

Appendix A List of Acronyms

140A-TE	Vendor-documentation-dependent Test Evidence
ACMVP/AMVP	Automated Cryptographic Module Validation Project
ACVP	Automated Cryptographic Validation Protocol
AD DS	Active Directory Domain Services
ALB	Application Load Balancer
API	Applications Programming Interface
AS	Assertion
CAVP	Cryptographic Algorithm Validation Program
CCCS	Canadian Centre for Cyber Security
CL	Component List
CMVP	Cryptographic Module Validation Program
CRADA	Cooperative Research and Development Agreement
CSTL	Cryptographic and Security Testing Laboratory
CVE	Common Vulnerabilities and Exposures
DMS	Database Migration Service
ECR	Elastic Container Registry
ECS	Elastic Container Service
EDC	Error Detection Code
EFP	Environmental Failure Protection
EFT	Environmental Failure Testing
EKS	Elastic Kubernetes Service
ESV	Entropy Source Validation
FIPS	Federal Information Processing Standards
FSM	Finite State Model
FT	Functional Test
FW	Firmware
HW	Hardware
ICMC	International Cryptographic Module Conference
IEC	International Electrotechnical Commission
IG	Implementation Guidance
ISO	International Organization for Standardization
IUT	Implementation Under Test
MAC	Message Authentication Code
MIS	Module Information Structure
MQP	Message Queue Processor

NCCoE	National Cybersecurity Center of Excellence
NLB	Network Load Balancer
NVLAP	National Voluntary Laboratory Accreditation Program
OD	Other Documents
OTAR	Over the Air Rekeying
OTP	One-time Programmable
PAA	Processor Augmented Algorithm
PAI	Processor Algorithm Implementation
RDS	Relational Database Service
S3	Simple Storage Service
SC	Source Code
SP	Security Policy
SQL	Structured Query Language
SSP	Sensitive Security Parameter
SW	Software
TE	Test Evidence
VE	Vendor Evidence
WS	Workstream

Appendix B CMVP TEs Tables

Applicable TEs for each combination of the basic filtering criteria based on TETables_v2.3.03.json developed by the NCCoE ACMVP project team can be found on the ACVMP Documentation website.

Appendix C CMVP Demo Server

A Demo server was deployed at NIST enabling the community to explore and get acquainted with the newly developed application. Information on how to access the Demo server is available on the ACMVP Documentation website.

Appendix D Application Modernization

The Infrastructure Workstream researched containerization approaches, Containers and EC2 launch types, Elastic Load Balancers, and Authentication layers that led to the final iteration. The services explored are summarized in this appendix.

D.1 Microsoft Windows Containers

Microsoft Windows containers were the starting point of the research since they run the same OS as the legacy ACMVP infrastructure. Additionally, they allow the use of Microsoft IIS in the container to handle the mTLS handshake for authentication. The applications were successfully containerized and enabled the modernization of the supporting infrastructure. However, there was a limitation with the AWS CodeBuild/CodePipeline integration, which requires Docker-in-Docker.

D.2 Linux Containers

Linux containers do not support Microsoft IIS (where mTLS authentication is handled), which resulted in research for alternative authentication mechanisms. Microsoft-supported DotNet container images provide the necessary components to run the C# WebPublic application. nginx was selected as an open-source solution that provides mTLS configuration support and proxy capabilities.

D.3 Amazon EC2 Launch

This container launch type utilizes a base Amazon Machine Image (AMI) to launch onto an Amazon EC2 instance. The container runs via a Docker daemon and is built locally. Network connections are routed through the Amazon EC2 instance to the underlying container.

D.4 Amazon EC2 Fargate Launch

The serverless Amazon ECS Fargate service provides a hosted platform for containerized tasks and services. Managed components consist of automation around host provisioning and compute monitoring. The end user is responsible for managing Amazon ECS tasks or service definitions that interface with the AWS-provided host through a mixture of AWS Identity and Access Management (IAM) controls, Amazon VPC security groups, and Elastic Network Interface (ENI) allocations.

D.5 Amazon ECS with Amazon EC2 Instance Launch

The Amazon ECS with Amazon EC2 Instance launch type allows system administrators a more granular control of the underlying Amazon EC2 instance hosting the container. While identified as an option, Amazon EKS Auto Mode was prioritized to ensure alignment with NIST Security practices for production containerized applications.

D.6 Amazon Fargate and Amazon EKS Launch

The Amazon EKS launch type was identified during this research. The team explored this option in earnest following ICMC '25. As with the Amazon ECS Fargate launch type, the foundational pieces controlling container workloads are managed and maintained by AWS.

As previously mentioned, the Amazon EKS service provides an AWS-managed solution for containerized workloads, which leverages the automated host and load balancer provisioning, auto-scaling integration, and workload access behind the scenes with a mixture of Amazon ALB, EC2, and VPC services. Cluster owners will manage how defined services and containerized workloads will interface with the underlying host through security groups and ENI mappings. Developers will manage application deployments and service connections via AWS ALB and RDS.

AWS Fargate provides an AWS-managed solution for launching containerized workloads as scalable services. Workloads are abstracted away from compute resources while providing connection points for containers via NCCoE-managed AWS VPC and subnets.

D.7 Layer 3 Authentication

D.7.1 nginx Reverse Proxy

nginx is a reverse proxy that routes requests to the ACMVP server, similar to the use of Microsoft IIS in the WebPublic application. nginx supports mTLS authentication, allowing it to verify client certificates before forwarding requests. nginx in a Linux container maintains robust load balancing, security, and authentication capabilities similar to Microsoft IIS in a Windows container.

D.7.2 AWS Network Load Balancer (NLB)

An AWS Network Load Balancer (AWS NLB) was initially used to route traffic to the containerized application with Microsoft IIS. Experiments using AWS Application Load Balancer (ALB) were conducted to test handling both container/service-level routing and application-level authentication previously handled by Microsoft IIS.

The architecture changes finally settled on the implementation of AWS NLB to handle the passthrough of the mTLS handshake. Certificate authentication and management are still passed on to the application for authentication, authorization, and logging details as required.

D.7.3 Amazon API Gateway

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. This service allows for a one-to-one layer of connection

between the gateway and the ACMVP web app endpoints and enables the development team to provision, distribute, and revoke API keys as an alternative and modern form of authentication for each API request made to the server. In combination with other services like AWS Cognito, labs could manage their own credentials to further improve operational efficiency.

Appendix E Research Infrastructure

The ACMVP project was performed in a research AWS cloud at the NCCoE following AWS best practices and NIST security compliance requirements. Details on the NCCoE AWS research cloud are provided on the ACMVP Documentation website.

To ensure compliance with existing NIST guidance, a mapping exercise was performed on the NCCoE AWS research cloud. Details on the mappings are provided on the ACMVP Documentation website.

Mapping to NIST 800 53

Mapping to NIST 800 92