**GSA’s Server-based Certificate Validation Protocol (SCVP) test program (GSTP)**

Usage Guide

[This page intentionally left semi-blank]

Revision History

|  |  |
| --- | --- |
| Date | Notes |
| 2017-06-08 | First draft |
| 2017-07-13 | Edits |
| 2017-08-08 | Added notes about setting SCVP\_OUTPUT\_PATH environment variable and alternative vss.properties file location |
| 2017-08-11 | Merged in sections on Virtual Machines, artifact hosting. |

1 Overview 5

2 GSTP Components 6

2.1 Test SCVP client 6

2.2 Test SCVP client scripts and script generator 9

2.3 Test SCVP client script generator 9

2.4 Test SCVP client script runner 11

2.5 Test artifacts 14

2.6 Sample environment 15

2.7 Hosts file for sample environment 15

3 GSTP usage 18

3.1 Generating test scripts 18

3.2 Executing GSTP test cases 19

3.3 Reviewing logs 19

3.3.1 Summary Results 19

3.3.2 Client Log 19

3.3.3 Validation Failures Re-execution Script 20

3.3.4 Profile Evaluation Failures Re-execution Script 20

3.3.5 Artifacts 20

3.3.6 Debug 20

4 Deploying Artifacts 21

4.1 Local VMs 21

4.1.1 Tools VM 21

4.1.2 Artifact Hosting VM 21

4.2 AWS image 23

4.3 Artifact Archives 23

Appendix A - Bibliography 24

# Overview

This document provides an overview of the artifacts and utilities employed by the GSA SCVP Test Program (GSTP). The GSTP aims to confirm an SCVP responder is capable of providing accurate certification path validation results in environments with comparable complexity to the U.S. Federal PKI. The test materials do not facilitate confirmation that a product is conformant with all aspects of the Server-based Certificate Validation Protocol (SCVP) as defined in RFC 5005. Instead, conformance to the SCVP profiles identified for use by GSA [TREAS] is demonstrated.

The GSTP is composed of seven primary components that are used to exercise an SCVP responder under test (RUT).

* Test SCVP client
* Test SCVP client scripts
* Test SCVP client script generator
* Test SCVP client script runner (optional)
* Test artifacts, i.e., certificates and revocation information
* Sample environment for hosting certificates and revocation information
* Hosts file to resolve names hosted by sample environment

The test SCVP client is provided along with a set of scripts to cause the client to use test artifacts to interact with a RUT. The test client will emit several streams of information including a summary of test results, basic logging information regarding test client operation, scripts to facilitate re-testing scenarios that failed to yield the expected results, debugging information and, optionally, request and response files for analysis.

A script generator is supplied to generate scripts to drive the test SCVP client. The script generator emits scripts for each set of test artifacts with several variations for interacting with the RUT.

Three distinct sets of test artifacts will be used to test certification path development and certification path validation capabilities:

1. NIST’s Public Key Infrastructure (PKI) Interoperability Test Suite v2 (PKITSv2)
2. NIST’s Path Development Test Suite v2 (PDTSv2)
3. Mock Federal PKI (MFPKI)

All AIA and CRL DP URIs included in the test artifacts feature names that are not routable on the public Internet. A Linux virtual machine that hosts artifacts via HTTP server and OCSP responder instances is available, along with a host file that can be tailored for use in resolving names during certification path processing.

# GSTP Components

## Test SCVP client

The GSTP test client is based on an SCVP client available from GSA on Github at: <https://github.com/GSA/vss>. The GSTP client will also be available via Github at a TBD location. The command line parameters accepted by the client are as follows:

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Parameter Type** | **Description** |
| -h, --help | None | Show help message and exit |
| **Basic Logistics** | | |
| --scvp\_profile | {lightweight, long-term-record, batch} | Name of SCVP profile |
| -x, --expectSuccess | Boolean value {true, false} | Indicates whether success is expected when validating the --target\_cert. Defaults to true |
| -l, --logging\_conf | Full path and filename of log4j configuration file | Used to customize default logging behavior |
| -n, --test\_case\_name | String value | Friendly name of test case |
| -z, --signer\_certs | Path to directory to receive certificate(s) used to validate SCVP responses | Save signer certificates as read from a validation policy response to a specified directory then exit |
| --log\_all\_messages | None | Log all requests and responses to the artifacts log, not just those from failed tests. Off by default. |
| **Target Certificate Details** | | |
| -c, --target\_cert | Full path and filename of binary DER encoded certificate | Certificate presented to responder for validation; not used when  --scvp\_profile is set to batch, required otherwise |
| -b, --batch\_folder | Full path of folder containing binary DER encoded certificates | Certificates presented to responder for validation; used when --scvp\_profile is set to batch, not used otherwise |
| -t, --trust\_anchor | Full path and filename of binary DER encoded certificate | Certificate presented to responder as trust anchor to use for validation; omitted from request by default |
| --batch\_folder\_success | Full path of folder containing binary DER encoded certificates | Certificates presented to responder for validation; used when --scvp\_profile is set to batch, not used otherwise; all certificates are expected to validate successfully |
| --batch\_folder\_failure | Full path of folder containing binary DER encoded certificates | Certificates presented to responder for validation; used when --scvp\_profile is set to batch, not used otherwise; all certificates are expected to fail validation |
| **SCVP Request Details** | | |
| -v, --validation\_policy | Object identifier value expressed in dot notation form (i.e., 1.2.3.4.5) | Validation policy to include in request; default value is 1.3.6.1.5.5.7.19.1 |
| --wantBacks | One or more symbolic WantBack names {Cert, BestCertPath, RevocationInfo, PublicKeyInfo, AllCertPaths, EeRevocationInfo, CAsRevocationInfo} | WantBack value(s) to include in request; default is BestCertPath |
| **Certification Path Validation Algorithm Inputs** | | |
| -p, --certificate\_policy | One or more object identifiers expressed in dot notation form (i.e., 1.2.3.4.5) | Certificate policies to use as the user supplied policy set; omitted from request by default |
| --inhibitAnyPolicy | Boolean value {true, false} | Boolean value to use as inhibitAnyPolicy; omitted from request by default |
| --inhibitPolicyMapping | Boolean value {true, false} | Boolean value to use as inhibitPolicyMapping; omitted from request by default |
| --requireExplicitPolicy | Boolean value {true, false} | Boolean value to use as requireExplicitPolicy; omitted from request by default |

Logging output is written to a location identified by the SCVP\_OUTPUT\_PATH environment variable.

Generally, the client need not be interacted with directly to execute test cases. A set of scripts are provided that drive execution of test scenarios in a variety of contexts. However, prior to using the scripts, the test client itself must be configured to interact with the RUT. A configuration file must be edited to provide the URL of the SCVP interface and a key store must be updated to include keys necessary to verify the SCVP responses. The configuration file is named vss.properties and is located in the /usr/local/tomcat/conf folder. The table below shows the settings that must be modified for test purposes.

|  |  |  |
| --- | --- | --- |
| **Configuration element** | **Purpose** | **Example value** |
| VSS\_TRUSTSTORE\_SCVP\_SIGNER\_ISSUER\_LABEL | Provides label of SCVP responder’s certificate in the keystore | someresponder |
| VSS\_SCVP\_SERVER\_URI | Provides the URI to which SCVP requests are sent. | http://example.com/scvp |
| VSS\_SCVP\_DER\_ENCODE\_DEFAULTS | Determines whether the client DER encodes default fields (some responders require presence of fields the DER requires to be absent). | False |
| VSS\_SCVP\_TEST\_CLIENT | Governs custom test client behavior that is only appropriate in a test client | True |
|  |  |  |

Alternatively, the location of the vss.properties file can be provided as a Java system variable when the client is launched as shown below (which also shows temporarily reassigning the SCVP\_OUTPUT\_PATH environment variable for a single run):

SCVP\_OUTPUT\_PATH=/<some path>/SCVP\_OUTPUT\_PATH2 java -Dvss.configLocation=/<some path>/vss.properties -jar vss2.jar --scvp\_profile lightweight -n 4.1.1 -c /<some path>/ValidCertificatePathTest1EE.crt --wantBacks BestCertPath

Once the configuration file edits have been performed, the RUT’s certificate must be added to the keystore.ks file located in the /usr/local/tomcat/conf folder. If the RUT’s certificate is not handy and the RUT supports validation policy requests, the test client can be used to retrieve the certificate via the following command:

java –jar vss2.jar –s /path/to/receive/certificate.der

The certificate may then be imported into the keystore using

keytool -keystore /usr/local/tomcat/conf/vssTrustStore.jks -importcert -file /path/to/receive/certificate.der -alias someresponder

The test client will write logs to the location identified by the SCVP\_OUTPUT\_PATH environment variable.

## Test SCVP client scripts and script generator

During the execution of the GSTP, the test SCVP client will be executed hundreds of times. To simplify execution of the test cases, a set of scripts are provided that reference a target certificate or collection of target certificates and provide a set of appropriate command line parameters. These scripts can be modified for the environment in which the test client will be used. Scripts may be manually altered or regenerated to change paths to test artifacts, to change output folder location or to change the list of wantBacks.

## Test SCVP client script generator

The script generator utility takes the following command line parameters:

ScvpScriptGenerator v1.0.0 usage

-h [ --help ] Print usage instructions

-l [ --logging\_conf ] arg Logging configuration to support report generation

--pkits\_2048\_folder arg Folder containing PKITS 2048 edition (root of

Renamed folder containing 0, 1, 2, etc. folders and

all certificates)

--pkits\_4096\_folder arg Folder containing PKITS 4096 edition (root of

Renamed folder containing 0, 1, 2, etc. folders and

all certificates)

--pkits\_p256\_folder arg Folder containing PKITS p256 edition (root of

Renamed folder containing 0, 1, 2, etc. folders and

all certificates)

--pkits\_p384\_folder arg Folder containing PKITS p384 edition (root of

Renamed folder containing 0, 1, 2, etc. folders and

all certificates)

--pdts\_folder arg Folder containing PDTS edition

--mfpki\_folder arg Folder containing MFPKI edition

--mfpki\_ta arg File containing the MFPKI trust anchor

--output\_folder arg Folder to receive generated scripts

-l [ --logging ] arg Logging configuration for ScvpScriptGenerator

logging purposes

--want\_back arg List of OIDS in dot notation form (i.e., 1.2.3.4.5)

to be passes as --wantBacks to the SCVP client

The following script can be tailored to regenerate a full complement of scripts to support execution of the GSTP against a given SCVP responder:

./ScvpScriptGenerator --mfpki\_folder /<path>/MFPKI/EEs --mfpki\_ta /<path>/MFPKI/TAs/905F942FD9F28F679B378180FD4F846347F645C1.fake.der

--output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pdts\_folder /<path>/PDTS/Renamed --output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_2048\_folder /<path>/PKITS\_2048/Renamed

--output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_4096\_folder /<path>/PKITS\_4096/Renamed

--output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_p256\_folder /<path>/PKITS\_P256/Renamed

--output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_p384\_folder /<path>/PKITS\_P256/Renamed

--output\_folder /<path>/GSTP --want\_back BestCertPath --want\_back RevocationInfo

The resulting output will be a set of scripts, as listed below. For the MFPKI and each PKITSv2 edition, a script targeting the default SCVP validation policy will be emitted both with and without trust anchor inclusion in the request for each SCVP profile type. PDTS will receive similar, except no batch scripts are emitted for PDTS. Similarly, for the MFPKI and each PKITSv2 edition, a script targeting a non-default SCVP validation policy will be emitted for each profile type. PDTS will receive similar, except no batch script is emitted. Fifty-one scripts are generated in total.

* MFPKI\_DEFAULT\_OMIT\_TA\_batch.sh
* MFPKI\_DEFAULT\_OMIT\_TA\_lightweight.sh
* MFPKI\_DEFAULT\_OMIT\_TA\_longterm.sh
* MFPKI\_DEFAULT\_WITH\_TA\_batch.sh
* MFPKI\_DEFAULT\_WITH\_TA\_lightweight.sh
* MFPKI\_DEFAULT\_WITH\_TA\_longterm.sh
* MFPKI\_NON\_DEFAULT\_batch.sh
* MFPKI\_NON\_DEFAULT\_lightweight.sh
* MFPKI\_NON\_DEFAULT\_longterm.sh
* PDTS\_DEFAULT\_OMIT\_TA\_lightweight.sh
* PDTS\_DEFAULT\_OMIT\_TA\_longterm.sh
* PDTS\_DEFAULT\_WITH\_TA\_lightweight.sh
* PDTS\_DEFAULT\_WITH\_TA\_longterm.sh
* PDTS\_NON\_DEFAULT\_lightweight.sh
* PDTS\_NON\_DEFAULT\_longterm.sh
* PKITS\_2048\_DEFAULT\_OMIT\_TA\_batch.sh
* PKITS\_2048\_DEFAULT\_OMIT\_TA\_lightweight.sh
* PKITS\_2048\_DEFAULT\_OMIT\_TA\_longterm.sh
* PKITS\_2048\_DEFAULT\_WITH\_TA\_batch.sh
* PKITS\_2048\_DEFAULT\_WITH\_TA\_lightweight.sh
* PKITS\_2048\_DEFAULT\_WITH\_TA\_longterm.sh
* PKITS\_2048\_NON\_DEFAULT\_batch.sh
* PKITS\_2048\_NON\_DEFAULT\_lightweight.sh
* PKITS\_2048\_NON\_DEFAULT\_longterm.sh
* PKITS\_4096\_DEFAULT\_OMIT\_TA\_batch.sh
* PKITS\_4096\_DEFAULT\_OMIT\_TA\_lightweight.sh
* PKITS\_4096\_DEFAULT\_OMIT\_TA\_longterm.sh
* PKITS\_4096\_DEFAULT\_WITH\_TA\_batch.sh
* PKITS\_4096\_DEFAULT\_WITH\_TA\_lightweight.sh
* PKITS\_4096\_DEFAULT\_WITH\_TA\_longterm.sh
* PKITS\_4096\_NON\_DEFAULT\_batch.sh
* PKITS\_4096\_NON\_DEFAULT\_lightweight.sh
* PKITS\_4096\_NON\_DEFAULT\_longterm.sh
* PKITS\_P256\_DEFAULT\_OMIT\_TA\_batch.sh
* PKITS\_P256\_DEFAULT\_OMIT\_TA\_lightweight.sh
* PKITS\_P256\_DEFAULT\_OMIT\_TA\_longterm.sh
* PKITS\_P256\_DEFAULT\_WITH\_TA\_batch.sh
* PKITS\_P256\_DEFAULT\_WITH\_TA\_lightweight.sh
* PKITS\_P256\_DEFAULT\_WITH\_TA\_longterm.sh
* PKITS\_P256\_NON\_DEFAULT\_batch.sh
* PKITS\_P256\_NON\_DEFAULT\_lightweight.sh
* PKITS\_P256\_NON\_DEFAULT\_longterm.sh
* PKITS\_P384\_DEFAULT\_OMIT\_TA\_batch.sh
* PKITS\_P384\_DEFAULT\_OMIT\_TA\_lightweight.sh
* PKITS\_P384\_DEFAULT\_OMIT\_TA\_longterm.sh
* PKITS\_P384\_DEFAULT\_WITH\_TA\_batch.sh
* PKITS\_P384\_DEFAULT\_WITH\_TA\_lightweight.sh
* PKITS\_P384\_DEFAULT\_WITH\_TA\_longterm.sh
* PKITS\_P384\_NON\_DEFAULT\_batch.sh
* PKITS\_P384\_NON\_DEFAULT\_lightweight.sh
* PKITS\_P384\_NON\_DEFAULT\_longterm.sh

## Test SCVP client script runner

The following script can be used to execute all GSTP test cases when run from a folder containing the test SCVP client with all logs collecting in one location.

bash /<path>/MFPKI\_DEFAULT\_OMIT\_TA\_batch.sh

bash /<path>/MFPKI\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/MFPKI\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/MFPKI\_DEFAULT\_WITH\_TA\_batch.sh

bash /<path>/MFPKI\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/MFPKI\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/MFPKI\_NON\_DEFAULT\_batch.sh

bash /<path>/MFPKI\_NON\_DEFAULT\_lightweight.sh

bash /<path>/MFPKI\_NON\_DEFAULT\_longterm.sh

bash /<path>/PDTS\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/PDTS\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/PDTS\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/PDTS\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/PDTS\_NON\_DEFAULT\_lightweight.sh

bash /<path>/PDTS\_NON\_DEFAULT\_longterm.sh

bash /<path>/PKITS\_2048\_DEFAULT\_OMIT\_TA\_batch.sh

bash /<path>/PKITS\_2048\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/PKITS\_2048\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/PKITS\_2048\_DEFAULT\_WITH\_TA\_batch.sh

bash /<path>/PKITS\_2048\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/PKITS\_2048\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/PKITS\_2048\_NON\_DEFAULT\_batch.sh

bash /<path>/PKITS\_2048\_NON\_DEFAULT\_lightweight.sh

bash /<path>/PKITS\_2048\_NON\_DEFAULT\_longterm.sh

bash /<path>/PKITS\_4096\_DEFAULT\_OMIT\_TA\_batch.sh

bash /<path>/PKITS\_4096\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/PKITS\_4096\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/PKITS\_4096\_DEFAULT\_WITH\_TA\_batch.sh

bash /<path>/PKITS\_4096\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/PKITS\_4096\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/PKITS\_4096\_NON\_DEFAULT\_batch.sh

bash /<path>/PKITS\_4096\_NON\_DEFAULT\_lightweight.sh

bash /<path>/PKITS\_4096\_NON\_DEFAULT\_longterm.sh

bash /<path>/PKITS\_P256\_DEFAULT\_OMIT\_TA\_batch.sh

bash /<path>/PKITS\_P256\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/PKITS\_P256\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/PKITS\_P256\_DEFAULT\_WITH\_TA\_batch.sh

bash /<path>/PKITS\_P256\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/PKITS\_P256\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/PKITS\_P256\_NON\_DEFAULT\_batch.sh

bash /<path>/PKITS\_P256\_NON\_DEFAULT\_lightweight.sh

bash /<path>/PKITS\_P256\_NON\_DEFAULT\_longterm.sh

bash /<path>/PKITS\_P384\_DEFAULT\_OMIT\_TA\_batch.sh

bash /<path>/PKITS\_P384\_DEFAULT\_OMIT\_TA\_lightweight.sh

bash /<path>/PKITS\_P384\_DEFAULT\_OMIT\_TA\_longterm.sh

bash /<path>/PKITS\_P384\_DEFAULT\_WITH\_TA\_batch.sh

bash /<path>/PKITS\_P384\_DEFAULT\_WITH\_TA\_lightweight.sh

bash /<path>/PKITS\_P384\_DEFAULT\_WITH\_TA\_longterm.sh

bash /<path>/PKITS\_P384\_NON\_DEFAULT\_batch.sh

bash /<path>/PKITS\_P384\_NON\_DEFAULT\_lightweight.sh

bash /<path>/PKITS\_P384\_NON\_DEFAULT\_longterm.sh

The following Python code (provided as GSTPScriptRunner.py) can be used to run the scripts listed above with logs moved in between each script.

**import** glob2  
**from** optparse **import** OptionParser  
**import** os  
**from** os.path **import** join  
**import** signal  
**from** subprocess **import** PIPE, Popen  
**import** sys  
**from** time **import** gmtime, strftime  
  
BASH\_EXE = **"/bin/bash"**bash\_process = **None***# noinspection PyUnusedLocal***def** signal\_handler(signal\_param, frame\_param):  
 **if** bash\_process:  
 bash\_process.kill()  
 print(**'Killed GSTP test execution process'**)  
 sys.exit(0)  
  
  
**def** main():  
 parser = OptionParser()  
 parser.add\_option(**"-i"**, **"--inputFolder"**, dest=**"input\_folder"**, default=**""**,  
 help=**"Folder containing scripts to run"**)  
 parser.add\_option(**"-l"**, **"--logFolder"**, dest=**"log\_folder"**, default=**""**,  
 help=**"Folder containing logs to move"**)  
 parser.add\_option(**"-d"**, **"--destLogFolder"**, dest=**"dest\_log\_folder"**, default=**""**,  
 help=**"Folder containing logs to move"**)  
 parser.add\_option(**"-p"**, **"--product"**, dest=**"product"**, default=**""**,  
 help=**"Short name of product under test (for use in naming relocated log folders)"**)  
  
 (options, args) = parser.parse\_args()  
  
 signal.signal(signal.SIGINT, signal\_handler)  
 **global** bash\_process  
  
 log\_folder = options.log\_folder  
 orig\_dest\_log\_folder = options.dest\_log\_folder  
 product = options.product  
  
 **if** os.path.isfile(os.path.join(log\_folder, **'artifacts.csv'**)):  
 os.remove(os.path.join(log\_folder, **'artifacts.csv'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'results.csv'**)):  
 os.remove(os.path.join(log\_folder, **'results.csv'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'client.txt'**)):  
 os.remove(os.path.join(log\_folder, **'client.txt'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'validation\_failures.txt'**)):  
 os.remove(os.path.join(log\_folder, **'validation\_failures.txt'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'profile\_failures.txt'**)):  
 os.remove(os.path.join(log\_folder, **'profile\_failures.txt'**))  
  
 **if** options.input\_folder:  
 only\_files = glob2.glob(options.input\_folder + **'/\*.sh'**)  
  
 **for** filename **in** only\_files:  
 t = strftime(**"%Y%m%d%H%M%S"**, gmtime())  
 print(**"Started "** + filename + **" at "** + t)  
  
 bash\_command = BASH\_EXE + **" "** + join(options.input\_folder, filename)  
 bash\_process = Popen(bash\_command, shell=**True**, stdout=PIPE)  
 bash\_process.wait()  
  
 *# noinspection PyUnusedLocal* process = **None** dest\_log\_folder = os.path.join(orig\_dest\_log\_folder, product + **"\_"** +  
 os.path.splitext(os.path.basename(filename))[0] + **"\_"** + t)  
 os.mkdir(dest\_log\_folder)  
 **if** os.path.isfile(os.path.join(log\_folder, **'artifacts.csv'**)):  
 os.rename(os.path.join(log\_folder, **'artifacts.csv'**), os.path.join(dest\_log\_folder, **'artifacts.csv'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'results.csv'**)):  
 os.rename(os.path.join(log\_folder, **'results.csv'**), os.path.join(dest\_log\_folder, **'results.csv'**))  
  
 art\_files = glob2.glob(options.log\_folder + **'/artifacts\*.csv'**)  
 **for** art **in** art\_files:  
 **if** os.path.isfile(art):  
 os.rename(art, os.path.join(dest\_log\_folder, os.path.basename(art)))  
  
 res\_files = glob2.glob(options.log\_folder + **'/results\*.csv'**)  
 **for** res **in** res\_files:  
 **if** os.path.isfile(res):  
 os.rename(res, os.path.join(dest\_log\_folder, os.path.basename(res)))  
  
 **if** os.path.isfile(os.path.join(log\_folder, **'client.txt'**)):  
 os.rename(os.path.join(log\_folder, **'client.txt'**), os.path.join(dest\_log\_folder, **'client.txt'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'debug.txt'**)):  
 os.rename(os.path.join(log\_folder, **'debug.txt'**), os.path.join(dest\_log\_folder, **'debug.txt'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'validation\_failures.txt'**)):  
 os.rename(os.path.join(log\_folder, **'validation\_failures.txt'**), os.path.join(dest\_log\_folder,  
 **'validation\_failures.txt'**))  
 **if** os.path.isfile(os.path.join(log\_folder, **'profile\_failures.txt'**)):  
 os.rename(os.path.join(log\_folder, **'profile\_failures.txt'**), os.path.join(dest\_log\_folder,  
 **'profile\_failures.txt'**))  
  
 t2 = strftime(**"%Y%m%d%H%M%S"**, gmtime())  
 print(**"Completed "** + filename + **" at "** + t2)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

This script will run all available scripts in the designated folder. To refrain from running certain scripts, simply delete or move them. For example, if not testing non-default validation policies, remove all of the scripts with non-default in the name.

## Test artifacts

PKITSv2 and PDTSv2 are updates to the existing NIST test suites. PKITS was updated to add AIA and CRL DP extensions to avoid the need to make all artifacts available locally to the product being tested. Additionally, editions were prepared using alternative public key and hash algorithms. PDTS was updated to feature unexpired artifacts, to drop LDAP-centric tests and to use RSA 2048 keys with SHA256 (instead of RSA 1024 with SHA1). While these were generated to support the GSTP, the artifacts are suitable for testing any RFC5280 compliant certification path validation implementation.

|  |  |  |
| --- | --- | --- |
| **Test Suite** | **Public Key Details** | **Hash Algorithm** |
| PKITSv2 | RSA 2048 | SHA256 |
|
|
| RSA 4096 | SHA512 |
|
|
| EC p256 | SHA256 |
|
|
| EC p384 | SHA384 |
|
|
| PDTSv2 | RSA 2048 | SHA256 |
|
| MFPKI | As observed (mostly RSA 2048) | As observed (mostly SHA256) |

MFPKI artifacts are cloned from the FPKI and do not have uniformly long validity periods like PDTSv2 and PKITSv2. Some artifacts that are classified as “good” will expire over time. PITTv2 can be used to periodically spot check so expired artifacts can be removed from service and/or re-refreshed using PCP.

## Sample environment

A Linux virtual machine is available that features artifacts from the MFPKI, various PKITSv2 editions hosted using Apache httpd and OpenSSL’s OCSP responder capabilities. The environment is intended to facilitate dynamic path discovery and avoid the need to manually provide artifacts to the RUT as a prerequisite for testing certification path validation capabilities.

## Hosts file for sample environment

A sample hosts file for the URIs included in artifacts that comprise the MFPKI, various PKITSv2 editions and PDTS is below.

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 betty.pkits.test

192.168.1.101 invalidcertificates.gov

192.168.1.101 testserver.testcertificates.gov

192.168.1.101 testserver.invalidcertificates.gov

192.168.1.101 testcertificates.gov

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 betty-4096.pkits.test

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 betty-256.pkits.test

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 betty-384.pkits.test

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 certipath-crl-ldap.verisign.com.test

192.168.1.101 certipath-aia.verisign.com.test

192.168.1.101 www.fis.evincible.com.test

192.168.1.101 dir1.com-strong-id.net.test

192.168.1.101 ssp-crl.symauth.com.test

192.168.1.101 crlserver.orc.com.test

192.168.1.101 pki.treasury.gov.test

192.168.1.101 strong-auth.eop.gov.test

192.168.1.101 ocs1.com-strong-id.net.test

192.168.1.101 certipath-sia.symauth.com.test

192.168.1.101 pilot-tscp-aia.symauth.com.test

192.168.1.101 rootweb.managed.entrust.com.test

192.168.1.101 nfi3.eva.orc.com.test

192.168.1.101 aces.ocsp.identrust.com.test

192.168.1.101 aia1.ssp-strong-id.net.test

192.168.1.101 sia1.ssp-strong-id.net.test

192.168.1.101 ocsp.dimc.dhs.gov.test

192.168.1.101 ocsp.dhhs.gov.test

192.168.1.101 crl.pki.va.gov.test

192.168.1.101 ocsp.managed.entrust.com.test

192.168.1.101 ssp-sia.verisign.com.test

192.168.1.101 ocspaces.trustdst.com.test

192.168.1.101 certipath-crl.symauth.com.test

192.168.1.101 keys.eop.gov.test

192.168.1.101 sspdir.managed.entrust.com.test

192.168.1.101 ocsp.pki.va.gov.test

192.168.1.101 crl.gds.disa.mil.test

192.168.1.101 crl.gds.nit.disa.mil.test

192.168.1.101 crl-server.orc.com.test

192.168.1.101 crl3.digicert.com.test

192.168.1.101 pilot-tscp-sia.symauth.com.test

192.168.1.101 tstocs3.com-strong-id.net.test

192.168.1.101 ssp-aia-ldap.verisign.com.test

192.168.1.101 sbca2.safe-biopharma.org.test

192.168.1.101 ocsp.uspto.gov.test

192.168.1.101 ocsp1.ssp-strong-id.net.test

192.168.1.101 tstcdp3.com-strong-id.net.test

192.168.1.101 ocspaces.identrust.com.test

192.168.1.101 cdp1.com-strong-id.net.test

192.168.1.101 devpki.treas.gov.test

192.168.1.101 pki.strac.org.test

192.168.1.101 devldap.treas.gov.test

192.168.1.101 rootdir.managed.entrust.com.test

192.168.1.101 devx500.arc.nasa.gov.test

192.168.1.101 ldap01.dimc.dhs.gov.test

192.168.1.101 sia1.com-strong-id.net.test

192.168.1.101 gpo-crl.ois.gpo.gov.test

192.168.1.101 doesspocsp.managed.entrust.com.test

192.168.1.101 devpki.treasury.gov.test

192.168.1.101 pki.fti.org.test

192.168.1.101 pub.carillonfedserv.com.test

192.168.1.101 apps.identrust.com.test

192.168.1.101 crls.pki.state.gov.test

192.168.1.101 ldap-pte.identrust.com.test

192.168.1.101 ldap.treas.gov.test

192.168.1.101 pilot-tscp-crl.symauth.com.test

192.168.1.101 lc.nasa.gov.test

192.168.1.101 hhspkicrl.managed.entrust.com.test

192.168.1.101 ocsp.defence.gov.au.test

192.168.1.101 aia3.com-strong-id.net.test

192.168.1.101 ldap.pki.va.gov.test

192.168.1.101 sspldap.treas.gov.test

192.168.1.101 crl.boeing.com.test

192.168.1.101 certipath-aia.symauth.com.test

192.168.1.101 pki.tscplab.org.test

192.168.1.101 sspweb.managed.entrust.com.test

192.168.1.101 ldap.digicert.com.test

192.168.1.101 certrep.pki.state.gov.test

192.168.1.101 crl.identrust.com.test

192.168.1.101 orc-ds.orc.com.test

192.168.1.101 ndac.arc.nasa.gov.test

192.168.1.101 crl.global.lmco.com.test

192.168.1.101 nfi2.eva.orc.com.test

192.168.1.101 certstatus.strac.org.test

192.168.1.101 dir.boeing.com.test

192.168.1.101 dir1.ssp-strong-id.net.test

192.168.1.101 sureid-aia.symauth.com.test

192.168.1.101 servers.cmcf.state.il.us.test

192.168.1.101 pki-crl.symauth.com.test

192.168.1.101 crl-pte.identrust.com.test

192.168.1.101 s.symcb.com.test

192.168.1.101 ssp-crl-ldap.verisign.com.test

192.168.1.101 www.fis.evincibletest.com.test

192.168.1.101 igc.ocsp.identrust.com.test

192.168.1.101 crl.disa.mil.test

192.168.1.101 s.symcd.com.test

192.168.1.101 www.gpo-fbca-crls.ois.gpo.gov.test

192.168.1.101 certstatus.fti.org.test

192.168.1.101 cdp1.ssp-strong-id.net.test

192.168.1.101 ldap.icam.pgs-lab.com.test

192.168.1.101 demodoesspweb.managed.entrust.com.test

192.168.1.101 ocsp.external.lmco.com.test

192.168.1.101 www.illinois.gov.test

192.168.1.101 ldap.fpki.gov.test

192.168.1.101 dsspweb.managed.entrust.com.test

192.168.1.101 ssp-sia.symauth.com.test

192.168.1.101 crl.external.lmco.com.test

192.168.1.101 sbca2-test.safe-biopharma.org.test

192.168.1.101 www.usps.com.test

192.168.1.101 www.dcs.exostar.com.test

192.168.1.101 ssp3.eva.orc.com.test

192.168.1.101 ssp-sia-ldap.verisign.com.test

192.168.1.101 www.tscp.eads.com.test

192.168.1.101 ts-mobile-qca.aia.com-strong-id.net.test

192.168.1.101 ssp-crl.verisign.com.test

192.168.1.101 ocsp.pki.state.gov.test

192.168.1.101 ipki.uspto.gov.test

192.168.1.101 ocsp.nsn0.rcvs.nit.disa.mil.test

192.168.1.101 www.defence.gov.au.test

192.168.1.101 tscp-crl.symauth.com.test

192.168.1.101 ts-mobile-qca.ocsp.com-strong-id.net.test

192.168.1.101 pki.treas.gov.test

192.168.1.101 publicsector.ocsp.identrust.com.test

192.168.1.101 cacerts.digicert.com.test

192.168.1.101 ssp-ocsp.verisign.com.test

192.168.1.101 directory.ois.gpo.gov.test

192.168.1.101 pki.dimc.dhs.gov.test

192.168.1.101 tstaia3.com-strong-id.net.test

192.168.1.101 http.fpki.gov.test

192.168.1.101 pki.raytheon.com.test

192.168.1.101 tscp-aia.symauth.com.test

192.168.1.101 apps-stg.identrust.com.test

192.168.1.101 cdp3.com-strong-id.net.test

192.168.1.101 crl4.digicert.com.test

192.168.1.101 cacerts.test.digicert.com.test

192.168.1.101 ts-mobile-qca.crl.com-strong-id.net.test

192.168.1.101 hhspkiocsp.managed.entrust.com.test

192.168.1.101 nfimediumsspdir.managed.entrust.com.test

192.168.1.101 ssp-aia.symauth.com.test

192.168.1.101 ssp-aia.verisign.com.test

192.168.1.101 sureid-crl.symauth.com.test

192.168.1.101 dir.defence.gov.au.test

192.168.1.101 nfirootdir.managed.entrust.com.test

192.168.1.101 ocsp1.com-strong-id.net.test

192.168.1.101 public.ocsp.identrust.com.test

192.168.1.101 doesspweb.managed.entrust.com.test

192.168.1.101 ldap.identrust.com.test

192.168.1.101 http.cite.fpki-lab.gov.test

192.168.1.101 certipath-aia-ldap.verisign.com.test

192.168.1.101 dsspdir.managed.entrust.com.test

192.168.1.101 crl.nit.disa.mil.test

192.168.1.101 certdata.northropgrumman.com.test

192.168.1.101 validation.identrust.com.test

192.168.1.101 nfimediumsspweb.managed.entrust.com.test

192.168.1.101 ocsp.disa.mil.test

192.168.1.101 certipath-crl.verisign.com.test

192.168.1.101 nfiocsp.managed.entrust.com.test

192.168.1.101 dsspocsp.managed.entrust.com.test

192.168.1.101 nfirootweb.managed.entrust.com.test

192.168.1.101 ocsp.treas.gov.test

192.168.1.101 apps-pte.identrust.com.test

192.168.1.101 hc.nasa.gov.test

192.168.1.101 pilot-certipath-aia.verisign.com.test

192.168.1.101 ocsp.digicert.com.test

192.168.1.101 www.ocsp.gpo.gov.test

192.168.1.101 ssp-ocsp.symauth.com.test

192.168.1.101 dir.tscp.eads.com.test

192.168.1.101 aia1.com-strong-id.net.test

192.168.1.101 tscp-sia.symauth.com.test

192.168.1.101 http.icam.pgs-lab.com.test

192.168.1.101 onsitecrl.verisign.com.test

192.168.1.101 eid-aia.symauth.com.test

192.168.1.101 pki-ocsp.symauth.com.test

192.168.1.101 ocsp.northropgrumman.com.test

192.168.1.101 pub.carillon.ca.test

192.168.1.101 igcrootpte.ocsp.identrust.com.test

192.168.1.101 ssp4.eva.orc.com.test

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\* Hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

192.168.1.101 betty.nist.gov.test

192.168.1.101 smime2.nist.gov.test

# \*\*\*\*\*\*\*\*\*\* End of hosts added by PCP VM preparation scripts \*\*\*\*\*\*\*\*\*\*

# GSTP usage

## Generating test scripts

Use the script generator to generate test scripts targeting the desired artifact collection. The example below demonstrates generation of test scripts targeting all six artifact collections located in /home/user/gstp with scripts written to /home/user/test. In this example, two wantBacks will be requested in each SCVP request (except batch, for which the script generator automatically omits all wantBacks).

./ScvpScriptGenerator --mfpki\_folder /home/user/gstp/MFPKI/EE\_good --mfpki\_ta/home/user/gstp/MFPKI/TAs/Common Policy-905F942FD9F28F679B378180FD4F846347F645C1.fake --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pdts\_folder /home/user/gstp/PDTS/renamed --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_2048\_folder /home/user/gstp/PKITS\_2048/renamed/ --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_4096\_folder /home/user/gstp/PKITS\_4096/renamed --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_p256\_folder /home/user/gstp/PKITS\_p256/renamed --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

./ScvpScriptGenerator --pkits\_p384\_folder /home/user/gstp/PKITS\_p384/renamed --output\_folder /home/user/test --want\_back BestCertPath --want\_back RevocationInfo

Delete any test scripts that are not of interest. For example, if not testing non-default validation policies, delete those scripts.

## Executing GSTP test cases

The RUT must be configured with all necessary trust anchors, any non-default validation policies and with the hosts file targeting the hosting environment that will be used. The test client must be configured to interact with the responder (in the vss.properties file) and save logs to an appropriate location (via the SCVP\_OUTPUT\_PATH environment variable).

After the RUT and client are configured, simply execute the desired test scripts and review the results. Make sure to delete any output files prior to test execution, if desired, because output files will be appended to throughout execution. The test runner script can be used to handle log file management.

## Reviewing logs

The test SCVP client is configured to emit six log streams, as described in the sub-sections below.

### Summary Results

The summary results file is written to results.csv and contains a brief summary of test SCVP client execution. It includes friendly name for a test, the expected result, an indication of expected result achieved and an indication of profile conformance evaluation.

### Client Log

The client log is written to client.txt. It contains additional detail not presented in the summary results. For example, an indication of which fields in an SCVP response caused profile conformance evaluation failure.

### Validation Failures Re-execution Script

The validation failures re-execution script is written to validation\_failures.txt. It includes invocations of the test SCVP client to enable re-execution of test cases that failed to yield the expected result with regard to validation of the target certificate(s).

### Profile Evaluation Failures Re-execution Script

The profile failures re-execution script is written to profile\_failures.txt. It includes invocations of the test SCVP client to enable re-execution of test cases that failed to yield the expected result with regard to evaluation of the SCVP response against the target SCVP profile.

### Artifacts

Base64 encoded SCVP requests and responses corresponding to failed test cases are written to artifacts.csv to facilitate detailed analysis using a utility like dumpasn1. To capture all request and responses, pass the --log\_all\_messages flag to the client.

### Debug

A debug log that includes all of the above plus lower level library output is emitted to aid in troubleshooting. The log information emitted by lower level libraries may include details that are not propagated back to the test client.

# Deploying Artifacts

The test PKI artifacts are supplied in three forms:

* Compressed archives containing only the artifacts themselves
* Installed on Virtual Machines (VMs) intended to be deployed locally
* Installed on an Amazon Web Services (AWS) image, suitable for deployment as an Elastic Compute Cloud (EC2) instance.

## Local VMs

Two Virutal Machines intended for local use are supplied. One includes the PKI Copy and Paste (PCP) tool used to generate the artifacts along with assorted other tools suitable for inspecting and testing them. It also includes a copy of the test harness and related utilities, ready to run to test an SCVP service.

The second VM includes a copy of all the artifacts and software configured to host them at the locations referenced in the certificates and service OCSP responses.

### Tools VM

The SCVP Tools VM is supplied in OVF format. In addition to being used to generate artifacts, it can be used to administer the artifacts VM.

* User: pcpadmin
* Password: aqswdefr1234!

The Tools VM should be deployed to the same virtual network as the artifact hosting VM. The pcpadmin user has an ssh keypair installed which can be used to administer the artifact hosting VM.

Tools included:

* All scripts and clients referenced in this guide, including copies of the sample hosts files that match the artifacts hosted in the Host VM, EC2 and Linode
* PCP itself
* FileZilla, for managing files on the hosting VM
* Firefox web browser
* Publication scripts referenced in the artifact publication guide
* Graphical TACT tools for editing settings databases and Trust Anchor stores
* openssl
* dumpasn1
* Xca
* Python 2 and Python 3
* PyCharm Community Edition

### Artifact Hosting VM

The artifact hosting VM supplied in OVF format. It is intended to be run without a graphical interface of any kind. It needs to have one interface on the same network as the Tools VM and one interface on the same network as the RUT. The hosts file installed on the RUT must reference the IP address of this network interface.

Once the Artifact Hosting VM is connected, log into the console using the following credentials:

* User: vmadmin
* Password: aqswdefr1234!

Use the ip command to find the current address of the network adapter connected to the RUT network:

[vmadmin@ficam-artifacts ~]$ ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 00:0c:29:be:31:f5 brd ff:ff:ff:ff:ff:ff

inet 192.168.99.210/24 brd 192.168.99.255 scope global dynamic ens33

valid\_lft 1547sec preferred\_lft 1547sec

inet6 fe80::f090:3a80:a0f2:b37c/64 scope link

valid\_lft forever preferred\_lft forever

3: ens34: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 00:0c:29:be:31:ff brd ff:ff:ff:ff:ff:ff

inet 10.142.42.2/24 brd 10.142.42.255 scope global ens34

valid\_lft forever preferred\_lft forever

inet6 fe80::20c:29ff:febe:31ff/64 scope link

valid\_lft forever preferred\_lft forever

And update the addresses in the hosts file installed on the RUT accordingly.

Start the httpd service by running the command

# systemctl start httpd

If the RUT will be using OCSP as well as CRLs to check status, open a command prompt in /srv/ocsp and run the command

# bash startall.sh

## AWS image

The AWS image is functionally identical to the local VM. Responders installed in the same AWS cloud can use its private IP address to access artifacts. Responders installed elsewhere can be added to the scvp-artifact-hosting security group and use the public IP of the ficam-scvp-artifacts VM.

## Artifact Archives

Artifacts are also supplied in zip archives within the Tools VM as well as published to the ficam-scvp-testing GitHub repository. These can be loaded into the SCVP responders per the vendor documentation for doing so.

# Appendix A - Bibliography

|  |  |
| --- | --- |
| [RFC5055] | Freeman, T., Housley, R., Malpani, A., Cooper, D. and W. Polk, "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, December 2007. |
| [TREAS] | Treasury Validation Services: SCVP Request and Response Profile, October 7, 2016. |