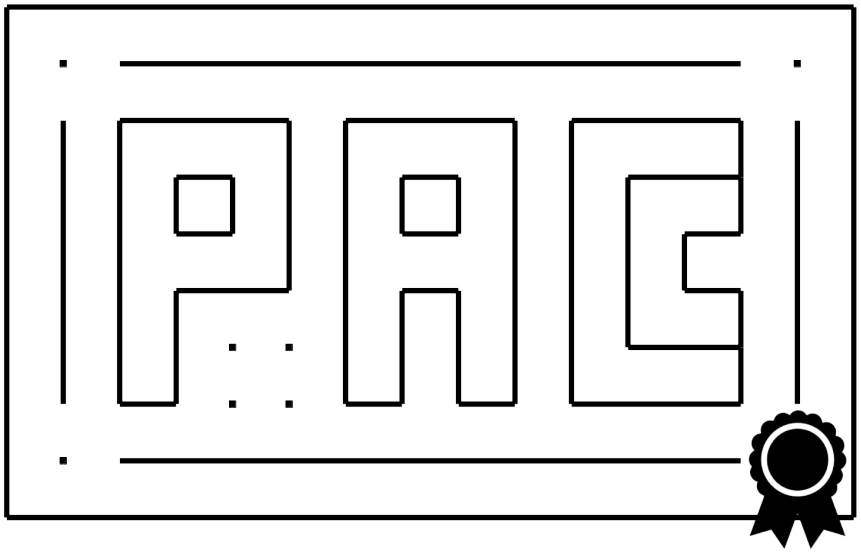

Platform Attribute Certificate Creator

Table of Contents

1 Quick Start

These tools target the TCG Platform Certificate Profile v1.0 [1].

The program consists of 3 tools. Each have usage details available on the command line via -h. The program requires Java 1.8 or greater. Only certain scripts have platform dependencies. Those dependencies are stated in the Scripts section.

Tools:
signer – Accepts observer details and data from an authority to produce a signed attribute certificate
validator – Performs a simple verification of the signature on an attribute certificate
observer – Gathers all data that should be set per host- local platform, components, properties details

The instructions here will guide the user to building and using the tools. Additional details are available later in the document.

Get started:

Build
(a) gradle clean build buildRpm
Install
(b) yum install ./build/distributions/paccor-*.rpm
(c) cd /opt/paccor
Gather data
(d) Run bash script
- ./scripts/allcomponents.sh > componentsFile

(e) Edit referenceoptions.sh for any variables necessary. See <link to section> for examples
(f) Run bash script
- ./scripts/referenceoptions.sh > optionsFile

(g) Run bash script
- ./scripts/get_ek.sh > ekFile

Define CA settings
(h) Edit otherextensions.sh for any variables necessary. See <link to section> for examples.
(i) Run bash script
- ./scripts/otherextensions.sh > extensionsFile

(j) Determine the new certificate serial number
(k) Determine the new certificate valid not before date
- Use the format YYYYMMDD

(l) Determine the new certificate valid not after date
- Use the format YYYYMMDD

(m) Know the path to your signing key and associated public key certificate, or make new ones
Create your certificate
(n) ./bin/observer -c componentsFile -p optionsFile -e ekFile -f observerFile
(o) ./bin/signer -o observerFile -x extensionsFile -b notBeforeDate -a notAfterDate -N serialNumber -k signingKey -P publicKeyCertificate -f signedCertificate
Verify the signature
(p) ./bin/validator -X x509v2AttributeCertificate -P publicKeyCertificate

2 Introduction

This program can create platform credentials according to the Platform Certificate Profile v1.0. A platform certificate (PC) is a X.509v2 Attribute Certificate which encapsulates details about components on a host and the security standards met by the platform manufacturer. The PC is signed by the platform manufacturer to prove its claims are authentic.

This program can assist in gathering all of the data that can go into a PC and produce a signed attribute credential. The tools within this program separate the collection of data destined for a PC into three categories.

1) **Local host component data** – Some amount of the data required for a PC is specific to each end-user device.

A script is provided to collect hardware descriptors and place them into a format which can be processed by the signer program. One main goal of this script is to provide an example of how to access those descriptors in a repeatable method. Anyone reading a platform certificate must be able to perform arbitrary verification that the certificate data reflects the device state. This means that the machine must respond in the same way when the user requests information as when the platform certificate was created.

The capability of the script is currently limited to CentOS 7. Support for additional platforms is forthcoming. The reason the script is limited by platform is that the commands to access hardware description fields are platform-specific, as are the commands to parse the responses.

2) **Host policy reference data** – The PC contains statements regarding standards which the platform manufacturer claims they met when they built the host. Most of these cannot be gathered from system commands. They are set on a per-host basis by the platform manufacturer.

A script is provided within which variables can be set by the manufacturer to fill in all relevant PC fields.

Use case: In order to support bulk processing of new platform certificates, the manufacturer might have a set of platform certificates which use the same policy reference script. If greater dynamic support is required, the script could be updated by the manufacturer to be informed by additional files or clues on the host.

3) **Other Extensions data** – There are additional extensions which must or should be included in new Platform Certificates. The data which goes into these extensions are relevant to a certificate or signing authority.

There is a script provided, similar to the host policy reference data, which will take values assigned to variables and output the extensions in a format which can be processed by the signer program.

The program consists of 3 tools. There is a brief description of each tool in the Quick Start section above. Details are provided in the sections below.

Note: In the usage output for each tool, parameters with an asterisk are required. Certificates and keys can be provided in DER or PEM format. The program will automatically handle either.

3 Signer

3.1 Description

This tool creates the X.509 v2 Attribute Certificate according to the Platform Attribute Certificate Profile. It will sign the certificate with the given private key. Data requested from the profile can be given to the tool via JSON. The JSON Structures section describes the format. Scripts are provided to reduce that work to only filling in a few variables. The signing algorithm is chosen by the signing public key certificate. See Supported Signing Algorithms.

3.2 Command Line Arguments

Arg	Alternate Arg	Format
-c	--componentJsonFile	See Component JSON
	the path to the file with the Device Info JSON	Use allcomponents.sh
-p	--policyRefJsonFile	See Policy Reference JSON
	the path to the file with the Policy Reference JSON	Use referenceoptions.sh
-e	--ekCertificate	Formatted in DER or PEM
	the path to a X.509 v3 EK Certificate	Use get_ek.sh
-N	--serialNumber	Integer
	serial number for the new certificate	
-b	--notBeforeDate	YYYYMMDD
	valid not before date	
-a	--notAfterDate	YYYYMMDD
	valid not after date	
-k	--privateKeyFile	Formatted in DER or PEM
	the private key used for signing the certificate	
-P	--publicCertificate	Formatted in DER or PEM
	the public key certificate corresponding to the private key	
-x	--extensionsJsonFile	See Extensions JSON

	the path to the file with the Extensions JSON	Use otherextensions.sh
-f	--out	Optional field
	the output file path. If not set, stdout will be used.	
	--pem	Optional field
	change output to PEM format. DER format is default	
-h	--help	
	prints the help message to stdout	

4 Validator

4.1 Description

This tool will validate the signature of a X.509 v2 Attribute Certificate. In the future, this validator could be enhanced to verify other claims within the attribute certificate.

4.2 Command Line Arguments

Arg	Alternate Arg	Format
-P	--publicCertificate	Formatted in DER or PEM
	the public key certificate of the signing key	
-X	--attributeCertificate	Formatted in DER or PEM
	the X.509v2 Attribute Certificate with the signature to validate	
-h	--help	
	prints the help message to stdout	

5 Observer

5.1 Description

The intention of this device observer tool is to provide a simple way to gather certificate data locally from a device and transfer it to a central location where the signer tool can perform its job. The output from the observer tool can be given directly to the signer.

5.2 Command Line Arguments

Arg	Alternate Arg	Format
-c	--componentJsonFile	See Component JSON
	the path to the file with the Component JSON	Use allcomponents.sh
-p	--policyRefJsonFile	See Policy Reference JSON
	the path to the file with the Policy Reference JSON	Use referenceoptions.sh
-e	--ekCertificate	Formatted in DER or PEM
	the path to a X.509 v3 EK Certificate	Use get_ek.sh
-f	--out	Optional field
	the output file path. If not set, stdout will be used.	

6 Scripts

6.1 allcomponents.sh

This script queries the operating system to find facts regarding hardware components on the device. The script should require no user interaction beyond running it.

6.1.1 Platforms Supported

- CentOS 7
  - Required to be run as a superuser
    - The OS commands called by the script require superuser privileges.
  - Required yum packages:
    - dmidecode
    - lshw

6.2 referenceoptions.sh

This script condenses Platform Attribute Certificate data fields into a few variables. The user should open the script and edit the variables for their purpose. Each variable below will reference the profile [1] for additional details.

Note: SHA-256 was chosen to be acceptable for each of the hashAlg choices for URI references. That can be changed by advanced users by editing this script.

6.2.1 Platforms Supported

- Any Unix distribution with bash
- Does not require special user privileges

6.2.2 Variable description

Variable	Value	Reference
tcgPlatformSpecificationMajorVersion	Number	TCG Platform Specification Attribute
tcgPlatformSpecificationMinorVersion	Number	
tcgPlatformSpecificationRevision	Number	
tcgCredentialSpecificationMajorVersion	Number	TCG Credential Specification Attribute
tcgCredentialSpecificationMinorVersion	Number	
tcgCredentialSpecificationRevision	Number	
platformConfigUri	URI	Platform Configuration Uri Attribute
platformConfigLocalCopyForHashing	Path	URI References require a hash of the file
tbbSecurityAssertionVersion	Number	Default: 1; Platform Assertions Attribute
commonCriteriaMeasuresVersion	String	see reference publications at https://CommonCriteriaPortal.org/cc
assuranceLevel	Number	valid options are 1 thru 7
evaluationStatus	Enum	valid options: designedToMeet, evaluationInProgress, evaluationCompleted
ccPlus	Enum	default false; valid options: true, false
strengthOfFunction	Enum	valid options: basic, medium, high
profileOid	ObjectId	OID of the protection profile
profileUri	URI	Platform Assertions Attribute
profileLocalCopyForHashing	Path	URI References require a hash of the file
targetOid	ObjectId	OID of the protection target

targetUri	URI	Platform Assertions
targetLocalCopyForHashing	Path	URI References require a hash of the file
fipsVersion	String	see reference publications at https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program/Standards
fipsLevel	Number	
fipsPlus	Enum	default false; valid options: true, false
measurementRootType	Enum	valid options: static, dynamic, nonHost, hybrid, physical, virtual
iso9000Certified	Enum	default false, valid options: true, false
iso9000Uri	URI	Simply a String in v1 of the spec.

6.3 otherextensions.sh

The Platform Attribute Certificate Profile [1] recommends the use of multiple certificate extensions. This script provides access to variables which can likely be set once by a certificate authority. In particular, the Certificate Policies, the CRL distribution list, and authority key access extensions can be defined using these variables. The user should open the script and edit the variables for their purpose. Each variable below will reference the profile [1] for additional details.

Note: SHA-256 was chosen to be acceptable for each of the hashAlg choices for URI references. That can be changed by advanced users by editing this script.

6.3.1 Platforms Supported

- Any Unix distribution with bash
- Does not require special user privileges

6.3.2 Variable description

Variable	Value	Reference
certPolicyOid1	ObjectId	Certificate Policies
certPolicyQualifierCPS1	String	Certificate Policies
certPolicyQualifierUserNotice1	String	Specific value defined in the profile
authorityInfoAccessMethod1	Enum	valid options are OCSP or CAISSUERS
authorityInfoAccessLocation1	DN	Authority Info Access
crlType	Number	valid options are 0 or 1
crlName	DN	CRL Distribution

crlReasonFlags	Number	valid options are integers 0 thru 16
crlIssuer	DN	CRL Distribution

6.4 get_ek.sh

This script will attempt to gather the EK certificate from a TPM.

6.4.1 Platforms Supported

- CentOS 7
  - Required to be run as a superuser
  - Should work for TPM v1.2 or v2.0

6.5 pc_certgen.sh

This script automates the entire process to use PACCOR with the goal of providing an example. It produces a platform certificate signed by a newly generated key pair. All settings can be customized. The generated key pair can be replaced with the user's specific key pair.

6.5.1 Platforms Supported

- CentOS 7
  - First run of the script required to be run as a superuser
    - This is in order to run allcomponents.sh and get_ek.sh.
    - Subsequent calls do not require superuser privileges

7 JSON Structures

7.1 Component JSON

```
{
    "PLATFORM": {
        "PLATFORMMANUFACTURERSTR": "",
        "PLATFORMMANUFACTURERID": "",   // optional
        "PLATFORMMODEL": "",
        "PLATFORMVERSION": "",
        "PLATFORMSERIAL": ""            // optional
    },
    "COMPONENTS": [
        {                               // zero or more component objects
            "MANUFACTURER": "",
            "MANUFACTURERID": "",       // optional
            "MODEL": "",
            "FIELDREPLACEABLE": "",     // optional
            "SERIAL": "",               // optional
            "REVISION": "",             // optional
            "ADDRESSES": [              // optional,
                                        //   zero or more of these 3 address types
                {
                    "ETHERNETMAC": ""
                },
                {
                    "WLANMAC": ""
                },
                {
                    "BLUETOOTHMAC": ""
                }
            ]
        }
    ],
    "PROPERTIESURI": {                  // optional
        "UNIFORMRESOURCEIDENTIFIER": "",// required if PROPERTIESURI is included
        "HASHALGORITHM": "",            // optional
        "HASHVALUE": ""                 // optional, base64 encode the value
    },
    "PROPERTIES": [                     // zero or more properties objects
        {
            "NAME": "",
            "VALUE": ""
        }
    ]
}
```

7.2 Policy Reference JSON

```
{
    "TCGPLATFORMSPECIFICATION": {
        "VERSION": {
            "MAJORVERSION": "",
            "MINORVERSION": "",
            "REVISION": "",
        },
        "PLATFORMCLASS": ""                         // base64 encode the class
    },
    "TCGCREDENTIALSPECIFICATION": {
        "MAJORVERSION": "",
        "MINORVERSION": "",
        "REVISION": ""
    },
    "TBBSECURITYASSERTIONS": {
        "VERSION": "",
        "ISO9000CERTIFIED": "",
        "CCINFO": {                                 // optional
            "VERSION": "",
            "ASSURANCELEVEL": "",                   // optional
            "EVALUATIONSTATUS": "",                 // optional
            "PLUS": "",
            "STRENGTHOFFUNCTION": "",               // optional
            "PROFILEOID": "",                       // optional
            "PROFILEURI": {                         // optional
                "UNIFORMRESOURCEIDENTIFIER": "",// required if PROFILEURI included
                "HASHALGORITHM": "",                // optional
                "HASHVALUE": ""                     // optional, base64 encode value
            },
            "TARGETOID": "",                        // optional
            "TARGETURI": {                          // optional
                "UNIFORMRESOURCEIDENTIFIER": "",// required if TARGETURI included
                "HASHALGORITHM": "",                // optional
                "HASHVALUE": ""                     // optional, base64 encode value
            }
        },
        "FIPSLEVEL": {                              // optional
            "VERSION": "",
            "LEVEL": "",
            "PLUS": ""
        },
        "RTMTYPE": "",                              // optional
        "ISO9000URI": ""                            // optional
    },
    "PLATFORMCONFIGURI": {                          // optional
        "UNIFORMRESOURCEIDENTIFIER": "",            // required if PLATFORMCONFIGURI
        "HASHALGORITHM": "",                        // optional
        "HASHVALUE": ""                             // optional, base64 encode value
    }
}
```

7.3 Extensions JSON

```
{
    "CERTIFICATEPOLICIES": [
        {
            "POLICYIDENTIFIER": "",
            "POLICYQUALIFIERS": [
                {
                    "POLICYQUALIFIERID": "",
                    "QUALIFIER": ""
                },{
                    "POLICYQUALIFIERID": "USERNOTICE",
                    "QUALIFIER": "TCG Trusted Platform Endorsement"
                }
            ]
        }
    ],
    "AUTHORITYINFOACCESS": [
        {
            "ACCESSMETHOD": "",
            "ACCESSLOCATION": ""
        }
    ],
    "CRLDISTRIBUTION": {
        "DISTRIBUTIONNAME": {
            "TYPE": "",
            "NAME": ""
        },
        "REASON": "",
        "ISSUER": ""
    }
}
```

7.4 Observer Output JSON

To be filled in

8 Supported Signing Algorithms

		OID
RSA	SHA1	1.2.840.113549.1.1.5
	SHA256	1.2.840.113549.1.1.11
	SHA384	1.2.840.113549.1.1.12
	SHA512	1.2.840.113549.1.1.13
DSA	SHA1	1.2.840.10040.4.3
	SHA256	2.16.840.1.101.3.4.3.2
	SHA384	2.16.840.1.101.3.4.3.3
	SHA512	2.16.840.1.101.3.4.3.4
ECDSA	SHA1	1.2.840.10045.4.1
	SHA256	1.2.840.10045.4.3.2
	SHA384	1.2.840.10045.4.3.3
	SHA512	1.2.840.10045.4.3.4

9 Glossary

ASN.1	Abstract Syntax Notation One
DER	Binary format of ASN.1 structures
EC	X509v3 Certificate representing the EK
EK	Endorsement Key
JSON	JavaScript Object Notation
PC	Platform Certificate
PEM	Base64 Encoded DER Certificate
TCG	Trusted Computing Group
TPM	Trusted Platform Module

10 References

[1] https://trustedcomputinggroup.org/resource/tcg-platform-attribute-credential-profile/