Metadata Bindings for sec-certs

Abstract

This document describes an approach/format proposed for bindings of third-party metadata files to relevant security certificates in sec-certs. The format should allow ideally for N:N bindings, with emphasis on extensibility, security, and usability aspects

High-level design and motivation

Design goals:

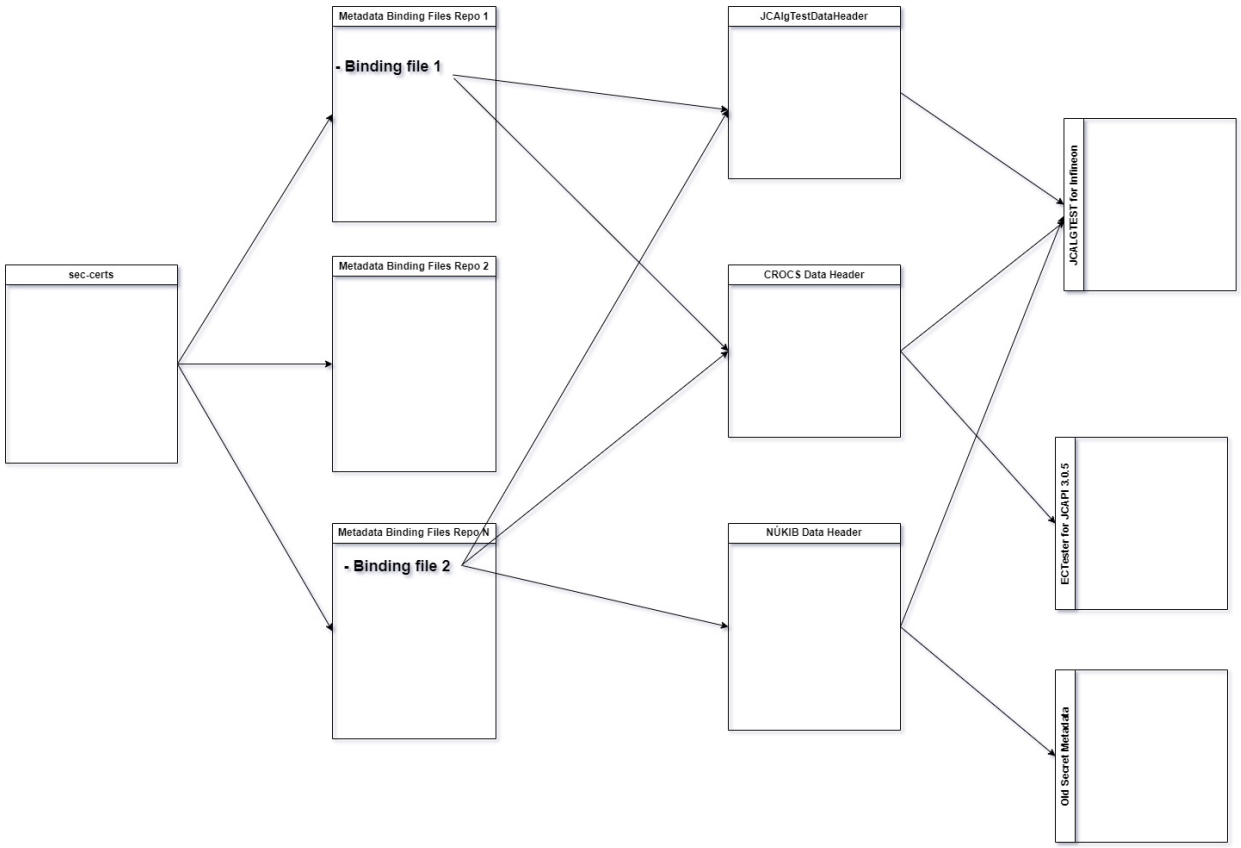
- Integrity of the metadata
- Integrity of the bindings and the metadata header files
- Separation of metadata from the sec-certs logic
    - We don't want to force metadata authors to create additional files and we want to be able to use virtually any metadata files
- Separation of the bindings and the metadata of the metadata files - header files
    - Support for old, unmaintained metadata files
- Human readability and editability
- Utilization of standard/proposed designs and technologies
- Minimization of complexity and administrator (repository) management
- Maximal extensibility for possible future adaptation

We decided that the "binding file" should be essentially split into two separate files. The first, is to be called the binding file as it really represents and creates the relationship. It does not directly reference the metadata, but rather a proxy file which is to be called the metadata header file. This metadata header file contains information about the metadata and a direct reference to them.

The primary motivation for this separation is the separation of the "responsibility" for the creation and authorship of the metadata and the binding itself. E.g. some authority may consider certain data relevant to specific certificates, while another authority may not which allows the user to choose depending on the level of trust to a certain author/ity. Another use case in which this design might be useful is when trying to create a binding with an older and unmaintained metadata file as the creation of the binding and the data measurements are now clearly distinguished.

Our primary concern are then the binding files "closer to sec-certs".

Bindings File Format



- *The newly proposed bindings file format is inspired by and utilizes the following standards and technologies. However, because of the absence of a format that would fully satisfy our requirements and fit our needs a new format was proposed.*
    - *https://datatracker.ietf.org/doc/rfc5544/ RFC inspiring the JSON format*
    - *https://www.rfc-editor.org/rfc/rfc7519?fbclid=IwAR00dHM3BLg1umIJY4KDADvRXOO2gDLfuHs676MlyJ4OYFPnr1BvI2ofyxk JWT RFC*
    - *https://datatracker.ietf.org/doc/html/rfc7515?fbclid=IwAR10DSgCnUH6_YS9MLmHRHTv5nPOcpJ_GkQN_coEsZ-I4DajYVX9jVQrkXs#appendix-A.4 JWS RFC*
- 0: An Object containing the "data" object, and the "JWT" and "author" strings
    - 1: "data": An Array of Binding Objects (Allows to express N:N relationships/bindings)
        - 2: Binding Object (Representing one binding of metadata file 1:N certificates)
            - Compulsory attributes: bindingAuthor, timestamp, certificateIDs, metadataHeaderURL
            - Required URL to the metadata file
            - 3: Required Array of Certificate IDs (Allows for 1:N relationship)
    - 1: "author": Author of the whole binding file
    - 1: "version": Version of the binding file
    - 1: "JWT": A JWT token of the {"data" : {}, "author": "", "version": ""} object

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
```

```
            "type": "object",
            "properties": {
                "data": {
                    "type": "array",
                    "items": [
                        {
                            "type": "object",
                            "properties": {
                                "timeStamp": {
                                    "type": "string"
                                },
                                "certificateIDs": {
                                    "type": "array",
                                    "items": [
                                        {
                                            "type": "string"
                                        }
                                    ]
                                },
                                "metadataHeaderURL": {
                                    "type": "string"
                                }
                            },
                            "required": [
                                "timeStamp",
                                "certificateIDs",
                                "metadataHeaderURL"
                            ]
                        },
                        {
                            "type": "object",
                            "properties": {
                                "timeStamp": {
                                    "type": "string"
                                },
                                "certificateIDs": {
                                    "type": "array",
                                    "items": [
                                        {
                                            "type": "string"
                                        },
                                        {
                                            "type": "string"
```

```
                                }
                            ]
                        },
                        "metadataHeaderURL": {
                            "type": "string"
                        }
                    },
                    "required": [
                        "timeStamp",
                        "certificateIDs",
                        "metadataHeaderURL"
                    ]
                }
            ]
        },
        "author": {
            "type": "string"
        },
        "version": {
            "type": "string"
        },
        "JWT": {
            "type": "string"
        }
    },
    "required": [
        "data",
        "author",
        "version",
        "JWT"
    ]
}
```

Metadata File Header Example Format

- 0: An Object containing the "data" object, and its "JWT"
  - 1: "data": An Array of objects representing the measurements, contains relevant metadata about the measurements and a reference to the metadata file
    - 2: Measurement Object
      - Compulsory attributes: metadataSHA256, timestamp, measurementTool, measurementAuthor, metadataURL
  - 1: "version": Version of the header file

- 1: "JWT": A JWT token of the {"data" : {}, "version": ""} object

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "data": {
            "type": "array",
            "items": [
                {
                    "type": "object",
                    "properties": {
                        "metadataSHA256": {
                            "type": "string"
                        },
                        "timeStamp": {
                            "type": "string"
                        },
                        "measurementTool": {
                            "type": "string"
                        },
                        "measurementAuthor": {
                            "type": "string"
                        },
                        "metadataURL": {
                            "type": "string"
                        }
                    },
                    "required": [
                        "metadataSHA256",
                        "timeStamp",
                        "measurementTool",
                        "measurementAuthor",
                        "metadataURL"
                    ]
                }
            ]
        },
        "version": {
            "type": "string"
        },
        "JWT": {
            "type": "string"
```

```
            }
        },
        "required": [
            "data",
            "version",
            "JWT"
        ]
    }
```

References

- In the proposed format, there are generally 2 types of references used to identify subjects of the relationship

1. The human readable IDs of security certificates in sec-certs - such as in the first column of seccerts.org list of results (SERTIT-116, NSCIB-CC-0031318-CR2, ...)
    - The certificates in sec-certs should be referenced in a human readable fashion
    - Therefore the human readable IDs - such as in the first column of seccerts.org list of results (SERTIT-116, NSCIB-CC-0031318-CR2, ...)
        - The exact ID format is still open for discussion as these IDs are inconsistent between certification authorities or could be insufficient in some cases
        - However it is not a goal of this document to properly define the ID format, but primarily it's extensibility and interchangability for future revisions is considered
    - *Disclaimer: This is in conflict with the inner representation in sec-certs which uses first bytes of a hash of several attributes of a certificate*
2. For referencing metadata files in their header files and for referencing these header files in the binding files we use standard URLs
    - This conviently satisfies the need for equal support of both local and network files/documents

Use Cases and Discussion

- Although virtually any binding files in the right format could be used and there can be many different repositories with the binding files, we also assume an existence of a maintained CROCS repository where such files could be found
- The metadataHeader files are primarily concern of either the author of the measurements or author of the bindings (can be a different person, often someone from the sec-certs team) that wants the data to be used
- After a discussion we concluded not to implement custom versioning of those binding files as it would result in significant, unnecessary complexity
    - In other words, the versioning shall be achieved by employing the mechanism in git

    - Invalidated files shall be simply replaced/removed also using standard git mechanisms and processes (which is for now the primary repository for those files)
- We will not thoroughly validate the metadataHeaders and referenced data - we will only trust and accept data signed by a trusted authority
    - We will not validate the header files, the referenced data or its sanity
    - We will not validate the data is really from the author or tool it claims to be
    - We should still validate at least: timestamps and hash of the metadata

Administrator Use Cases

- Deciding Pull Requests with new binding files (what should we validate?):
    - After a basic automated check of the format using GitHub actions, it shall also be reviewed by a human before it can be accepted
    - Validate JWT in the binding file
    - Validate URLs validity and availability of referenced files for each binding in the file
    - Validate timestamp (Is it in the past? Is it valid format?)
    - Validate the bindings is really from the author it claims to be (Use assymetric algorithm with JWT)
- Update of binding files:
    - By direct replacement/modification of the original file
        - What if the update changes author of the bindings? How do we trust it?
            - In such a case the file should not be modified but rather a new distinct binding file will exist
                - Could potentially lead to existence of multiple very similar files (not a concern for this document)
        - Need to reevalidate everything as with a new file
- Removal/Invalidation of binding files:
    - What is the lifespan of a binding file? Should it have one?
        - No, there will be no lifespan of the binding file
        - The file can still be removed from the repository (or manually removed from a local clone of the repository by a user)
    - Who can remove bindings from the repository?
        - Only the and administrators

Seccerts-User Use Cases

- The idea is the user can express which authorities he trusts
    - Only the metadata from trusted authors should be displayed
    - For the local development purposes, this could be simply just a matter of configuration (sources for the binding files) or deleting the unwanted binding files
    -
- Future proposition: In the web the user should be able to choose (filter) which authorities he trusts

- Potentially the tool performing the measurements could also be a usefull filter parameter