RATS Working Group
Internet-Draft
Intended status: Informational
Expires: 2 June 2024

D. Thaler
Microsoft
H. Birkholz
Fraunhofer SIT
T. Fossati
Arm

30 November 2023

RATS Endorsements
draft-ietf-rats-endorsements-00

Abstract

In the IETF Remote Attestation Procedures (RATS) architecture, a Verifier accepts Evidence and, using Appraisal Policy typically with additional input from Endorsements and Reference Values, generates Attestation Results in formats needed by a Relying Parties. This document explains the purpose and role of Endorsements and discusses some considerations in the choice of message format for Endorsements.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Remote ATtestation ProcedureS Working Group mailing list (rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/.

Source for this draft and an issue tracker can be found at https://github.com/dthaler/rats-endorsements.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Section 3 in the Remote ATtestation procedures (RATS) Architecture [RFC9334] gives an overview of the roles and conceptual messages in the IETF RATS Architecture. As discussed in that document, a Verifier accepts a well-defined set of RATS conceptual messages: Evidence, Endorsements and Reference Values (as well es Policy for Appraisal of Evidence). A Verifier appraises Evidence using Appraisal Policy for Evidence, typically against a set of Reference Values.

Various formats of conceptual messages exist, including standard and vendor-specific formats. One of the purposes of a Verifier is depicted in Figure 9 of [RFC9334]. A Verifier is intended to be able to accept Evidence in a variety of formats and generate Attestation Results in the formats needed by a Relying Parties it is intended to cater.

2. Actual State vs Reference States

Appraisal policies (Appraisal Policy for Evidence, and Appraisal Policy for Attestation Results) involve comparing the actual state of

an Attester against desired or undesired states, in order to determine how trustworthy the Attester is for its purposes. The state of an Attester represents its composition of components of execution environments (its "shape"), typically in a hierarchical manner. The state of an Attester also encompasses the combination of static and dynamic constitution (e.g., provisioned and deployed software, firmware, and micro-code), static and dynamic configuration, and the resulting operational state of its components at a certain point of time. Thus, a Verifier needs to receive messages with information about actual state, and information about desired/undesired states, and an appraisal policy that controls how the two are compared.

Each Attester in general has at least one Attesting Environment and one Target Environment (e.g., hardware, firmware, Operating System, etc.). Typically, each Attester has multiple Target Environments, each with their own set of claims (sometimes called a "claimset") representing their actual state. Additionally, the number of Target Environments is not limited.

"Actual state" is a group of claimsets about the actual state of the Attester at a given point in time. Each claimset holds claims about a specific Target Environment that is essential to determining trustworthiness. Generally speaking, each claim has a name (or other ID) and a singleton value, being the value of that specific Attester at a given point in time. Some claims may inherently have multiple values, such as a list of files in a given location on the device, but for our purposes we will treat such a list as a single unit, meaning one Attester at one point in time.

"Reference state" is a group of claimsets about the desired or undesired state of the Attester. Typically, each claim has a name (or other ID) and a set of potential values, being the values that are allowed/disallowed when determining whether to trust the Attester. In general there may be more gradation than simply "allowed or disallowed" so each value might include some more complex level of gradation in some implementations.

That is, where actual state has a single value per claim per Target Environment applying to one device at one point in time, reference state can have a set of values per claim per Target Environment. The appraisal policy then specifies how to match the actual value against the set of Reference Values.

Some examples of such matching include:

* The actual value must be in the set of allowed Reference Values.

* The actual value must not be in the set of disallowed Reference Values.

* The actual value must be in a range where two Reference Values are the min and max.

2.1. RATS Conceptual Messages

RATS conceptual messages in [RFC9334] fall into the above categories as follows:

* Actual state: Evidence, Endorsements, Attestation Results

* Reference state: Reference Values

* Appraisal policy: Appraisal Policy for Evidence, Appraisal Policy for Attestation Results

The figure below shows an example of Verifier input for a layered Attester as discussed in [RFC9334].

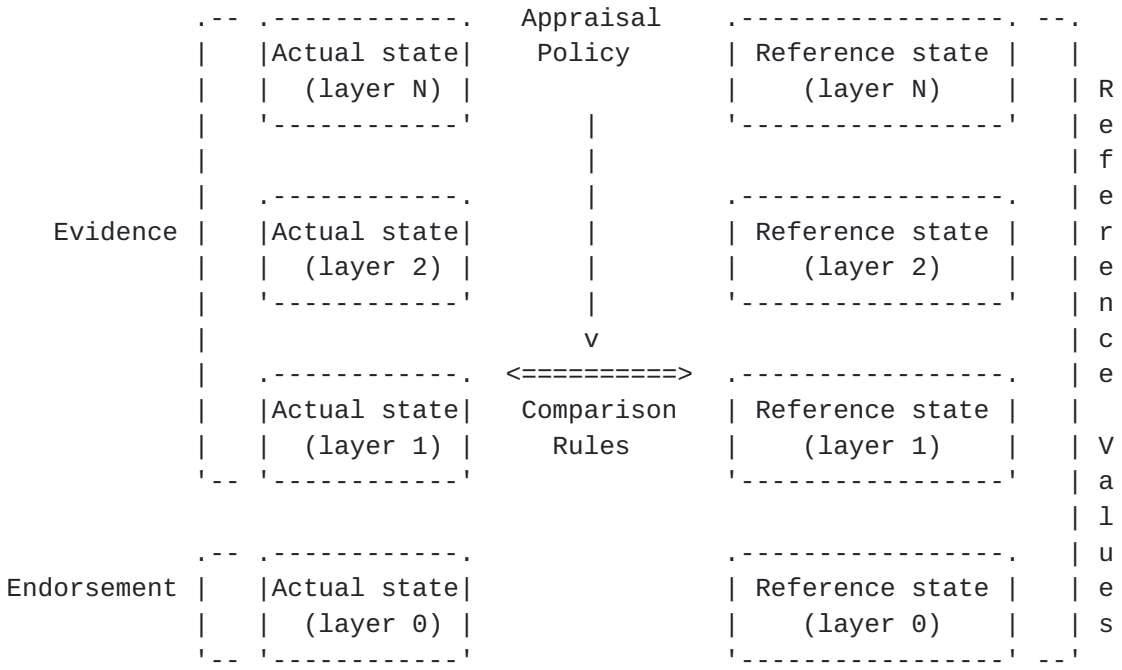

Figure 1: Example Verifier Input

While the above example only shows one layer within Endorsements as the typical case, there could be multiple layers (see Section 5), such as a chip added to a hardware board potentially from a different vendor.

A Trust Anchor Store is a special case of state above, where the Reference State would be the set of trust anchors accepted (or rejected) by the Verifier, and the Actual State would be a trust anchor used to verify Evidence or Endorsements.

In layered attestation using DICE [TCG-DICE] for example, the actual state of each layer is signed by a key held by the next lower layer.

Thus in the example diagram above, the layer 2 actual state (e.g., OS state) is signed by a layer 1 key (e.g., a signing key used by the firmware), the layer 1 actual state (e.g., firmware state) is signed by a layer 0 key (e.g., a hardware key stored in ROM), and the layer 0 actual state (hardware specs and key ID) is signed by a layer 0 key (e.g., a vendor key) which is matched against the Verifier's trust anchor store, which is part of the layer 0 reference state depicted above.

3. Conditionally Endorsed Values

Some claims in Endorsements might be conditional. A claim is conditional if it only applies if actual state matches Reference Values, according to some matching policy.

Endorsers should not use conditionally endorsed values based on immutable values of actual state in Evidence (such as an immutable serial number for example). An Endorser can, however, use conditionally endorsed values based on mutable values. For example an Endorser for a given CPU might provide additional information about what the CPU supports based on current firmware configuration state.

Policies around matching actual state in Evidence against reference states are normally expressed in Appraisal Policy for Evidence. Similarly, reference states are normally expressed in the Reference Values conceptual message. Such policies allow a Verifier and Relying Parties to make their decisions about trustworthiness of an Attester.

The use of conditionally endorsed values, however, is different in that a matching policy is not about trustworthiness (and hence not "appraisal" per se) but rather about whether an Endorser's claim is applicable or not, and thus usable as input to trustworthiness appraisal or not.

As such the matching policy for conditionally endorsed values must be up to the Endorser not the Appraisal Policy Provider. Thus, an Endorsement format that supports conditionally endorsed values would probably include some minimal matching policy (e.g., exact match against a singleton reference value). This unfortunately complicates design as a Verifier may need multiple parsers for matching policies.

4. Endorsing Identity

One type of claims that might be endorsed would be claims having to do with identity, such as verification keys. While identity claims are just another type of claims that may be endorsed, some implementations might treat them differently. For example, a Verifier might perform a first step to cryptographically verify the Attester's identity before spending effort on another step to

appraise other claims for determining trustworthiness.

This document treats identity claims as with any other claims, but allows Appraisal Policy for Evidence to have multiple steps if desired.

5. Multiple Endorsements

Figure Figure 1 showed an example with an Endorsement at layer 0, such as a hardware manufacturer providing claims about the hardware. However, the same could be done at other layers in addition. For example, an OS vendor might provide additional static claims about the OS software it provides, and application developers might provide additional static claims about the applications they release.

Figure 2 depicts an example with an Attester consisting of an application, OS, firmware, and hardware, each from a different vendor that provides an Endorsement for their own Target Environment, containing additional claims about that Target Environment. Thus each Target Environment (application, OS, firmware, and hardware) has one set of claims in the Evidence, and an additional set of claims in the Endorsement from its manufacturer. A Verifier that trusts each Endorser would thus use claims from both conceptual messages when comparing against reference state for a given Target Environment.

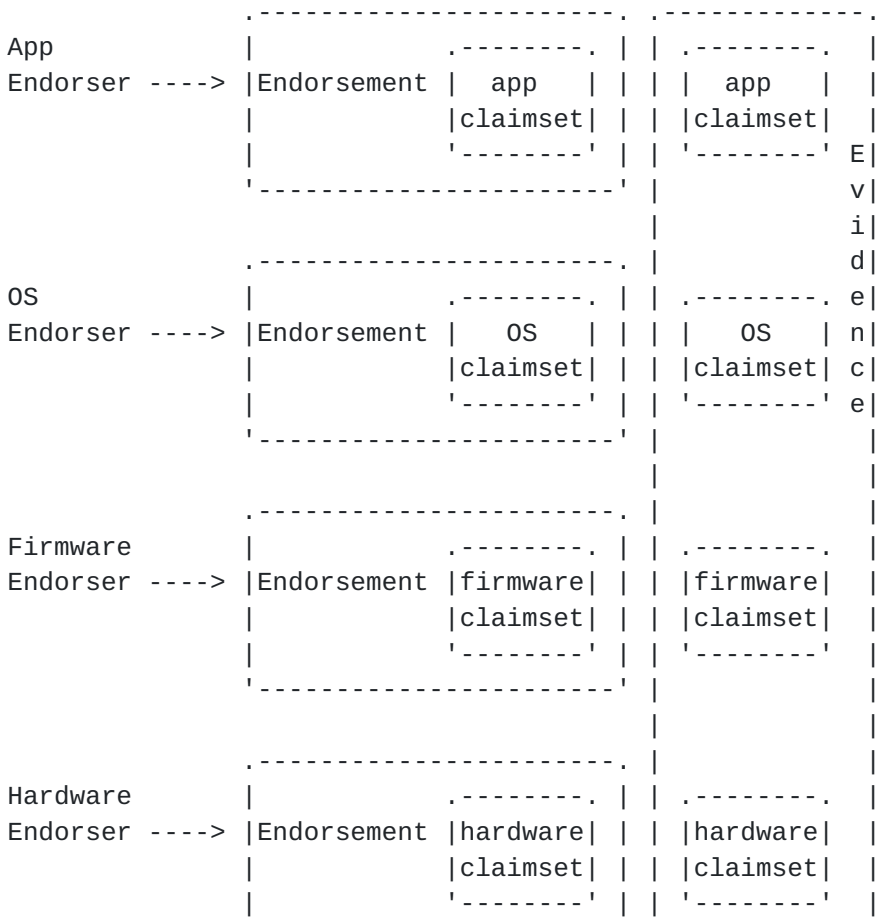

Figure 2: Multiple Endorsements

When Target Environments from different vendors each have their own Endorser, it is important that a Verifier be able to distinguish which Endorser is allowed to provide an Endorsement about which Target Environment. For example, the OS Endorser might be trusted to provide additional claims about the OS, but not about the hardware. Thus it is not as simple as saying that a Verifier has a trusted set of Endorsers. The binding between Target Environment and Endorser might be part of the Appraisal Policy for Evidence, or might be specified as part of the Evidence itself, or some combination of the two. An Endorsement format specification should explain how this concern is addressed.

6. Endorsement Format Considerations

This section discusses considerations around formats for Endorsements.

6.1. Security Considerations

In many scenarios, a Verifier can also support a variety of different formats, and while code size may not be a huge concern, simplicity and correctness of code is essential to security. "Complexity is the enemy of security" is a popular security mantra and hence to increase security, any decrease in complexity helps. As such, using the same format for both Evidence and Endorsements can reduce complexity and hence increase security.

6.2. Scalability Considerations

We currently assume that Reference Value Providers and Endorsers typically provide the same information to a potentially large number of clients (Verifiers, or potentially to other entities for later relay to a Verifier), and are generally on devices that are not constrained nodes, and hence additional scalability, including code size, is not a significant concern.

The scenario where scalability in terms of code size is strongest, however, is when a Verifier is embedded into a constrained node. For example, when a constrained node is a Relying Party for most purposes, but still needs a way to establish trust in the Verifier it will use. In such a case, the Relying Party may have a constrained Verifier embedded in it that is only capable of appraising Evidence provided by its desired Verifier. Thus, the Relying Party uses its embedded Verifier for purposes of appraising its desired Verifier

which it treats as only an Attester, and once verified, then uses it for verification of all other Attesters. In this scenario, the embedded Verifier may have code and data size constraints, and a very simple (by comparison) Appraisal Policy for Evidence and desired state (e.g., a required trust anchor that Evidence must be signed with and little else).

Using the same message format for Evidence, Endorsements, and (later) Attestation Results received from the later Verifier, can provide a code size savings due to having only a single parser in this limited case.

Similarly, an embedded constrained Verifier can choose to not support conditionally endorsed values, in order to avoid complexity introduced by such.

7. IANA Considerations

This document does not require any actions by IANA.

8. Acknowledgements

The authors wish to thank Thomas Hardjono, Laurence Lundblade, and Kathleen Moriarty for feedback and ideas that contributed to this document.

9. References

9.1. Normative References

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

9.2. Informative References

[TCG-DICE] Trusted Computing Group, "DICE Certificate Profiles", n.d., <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Certificate-Profiles-r01_3june2020-1.pdf>.

Authors' Addresses

Dave Thaler
Microsoft
United States of America
Email: dave.thaler.ietf@gmail.com

Henk Birkholz
Fraunhofer SIT

Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de

Thomas Fossati
Arm
Email: Thomas.Fossati@arm.com