

An EAT Profile for AR4SI



RATS WG, IETF 115, London

AR4SI recap

- RATS I-D draft-ietf-rats-ar4si
- Goal is to define an *info model* for conveying normalised attestation result
- The core construct is the trustworthiness vector:
 - 8x256 matrix of pre-defined + customisable semantics
 - set of rules for computing the vector's values

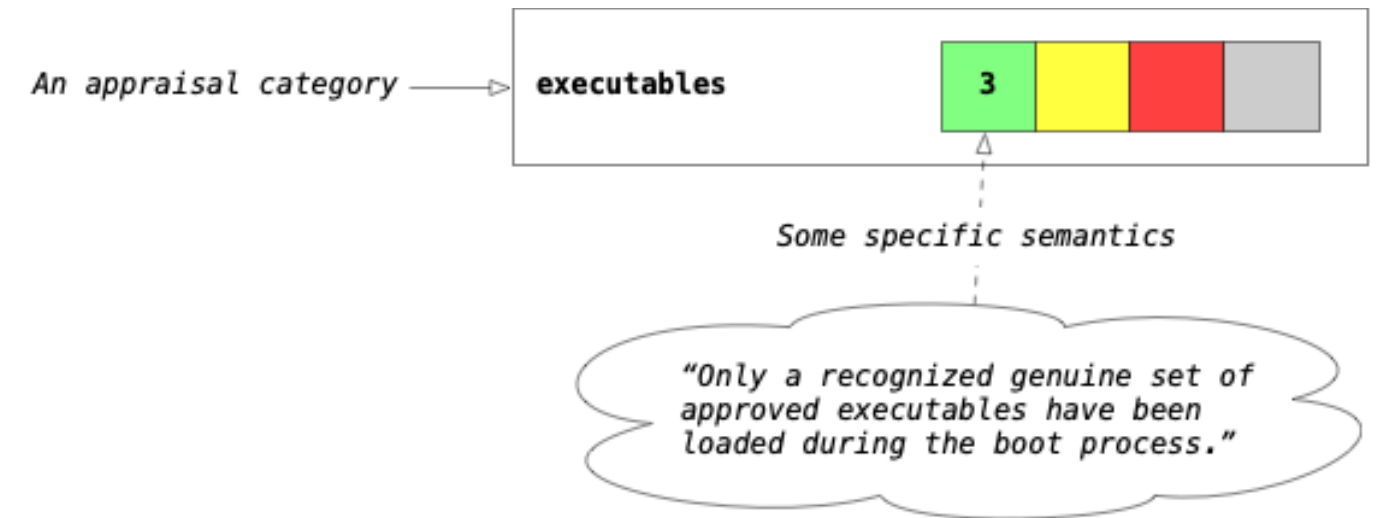
Trustworthiness Tiers

- y: 256 code-point space (8b signed int) organised in four tiers
- x: two sub-spaces (standard, private)

	std	private
affirming	2..31	-32..-2
warning	32..95	-96..-33
contraindicated	96..127	-128..-97
none	-1..1	

Trustworthiness Claim

Each "trustworthiness claim" is associated to an appraisal category and, for that category, the claim defines its own semantics.



Trustworthiness Vector

The “trustworthiness vector” is a collection of 8 pre-defined “trustworthiness claims”.

A missing entry is equivalent to 0 (i.e., no claim in this category).

Worth noting: the vector allows exposing the scope of conclusions (and in a sense the *quality* of the verifier.)

configuration	2
executables	3
file-system	0
hardware	2
instance-identity	2
runtime-opaque	32
sourced-data	0
storage-opaque	0

Info vs Data Model

AR4SI only provides the semantic core of the appraisal result.

However, a RP also needs other metadata, e.g.:

- identity of the verifier (e.g., cryptographic identity, software identity)
- time of the appraisal
- an identifier of the appraisal policy
- maybe evidence about the verifier's execution environment (e.g., in TEE)

Besides, AR4SI does not define a *data model*.

An EAT-based Serialisation

In Veraison have defined a serialisation:

```
ar4si-trustworthiness-vector = non-empty<{  
  ? instance-identity => $ar4si-trustworthiness-claim  
  ? configuration => $ar4si-trustworthiness-claim  
  ? executables => $ar4si-trustworthiness-claim  
  ? file-system => $ar4si-trustworthiness-claim  
  ? hardware => $ar4si-trustworthiness-claim  
  ? runtime-opaque => $ar4si-trustworthiness-claim  
  ? storage-opaque => $ar4si-trustworthiness-claim  
  ? sourced-data => $ar4si-trustworthiness-claim  

```

```
$ar4si-trustworthiness-claim = -128..127
```

And wrapped it into a top-level EAT Claims-Set called
EAR (EAT Attestation Result)

```
EAR = {  
  ear.status => $ar4si-trust-tier  
  eat_profile => "tag:github.com/veraison/ar4si,2022-10-17"  
  ? ear.trustworthiness-vector => ar4si-trustworthiness-vector  
  ? ear.raw-evidence => ear-bytes  
  iat => numeric-date  
  ? ear.appraisal-policy-id => text  
  * $$ear-extension  
}
```


JSON / JWT Example

```
{  
  "eat_profile": "tag:github.com/veraison/ar4si,2022-10-17",  
  "ear.status": "contraindicated",  
  "ear.trustworthiness-vector": {  
    "instance-identity": 32,  
    "configuration": 32,  
    "executables": 96,  
    "hardware": 2  
  },  
  "ear.appraisal-policy-id": "https://veraison.example/policy/1/60a0068d",  
  "iat": 1666529184  
}
```

CBOR / CWT Example

```
{
  265: "tag:github.com/veraison/ar4si,2022-10-17",
  1000: 96,
  1001: {
    0: 32,
    1: 32,
    2: 96,
    4: 2
  },
  1003: "https://veraison.example/policy/1/60a0068d",
  6: 1666529184
}
```

Veraison-specific Extensions

Plug into the `$$ear-extension` socket

→ Easy-to-consume breakdown of the evidence claims-set

```
ear-veraison-processed-evidence = {  
  + ear-label => any  
}
```

→ Any claim "derived" by the Verifier during appraisal (e.g., the certification status of a device)

```
ear-veraison-verifier-added-claims = {  
  + ear-label => any  
}
```

Example

```
{
  "eat_profile": "tag:github.com/veraison/ar4si,2022-10-17",
  "ear.status": "affirming",
  "ear.trustworthiness-vector": {
    "instance-identity": 2,
    "configuration": 2,
    "executables": 2,
    "hardware": 2
  },
  "iat": 1666529284,
  "ear.appraisal-policy-id": "https://veraison.example/policy/1/60a0068d",
```

Example (cont.)

```
[...]
"ear.veraison.processed-evidence": {
  "eat-profile": "http://arm.com/psa/2.0.0",
  "psa-client-id": 1,
  "psa-security-lifecycle": 12288,
  "psa-implementation-id": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
  "psa-software-components": [
    {
      "measurement-value": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
      "signer-id": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
    },
    {
      "measurement-value": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
      "signer-id": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
    }
  ],
  "psa-nonce": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
  "psa-instance-id": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyAh",
  "psa-certification-reference": "1234567890123-12345"
},
```

Example (cont.)

```
[...]
"ear.veraison.verifier-added-claims": {
  "psa-certified": {
    "certificate-number": "1234567890123-12345",
    "date-of-issue": "23/06/2022",
    "test-lab": "Riscure",
    "certification-holder": "ACME Inc.",
    "certified-product": "RoadRunner",
    "hardware-version": "Gizmo v1.0.2",
    "software-version": "TrustedFirmware-M v1.0.6",
    "certification-type": "PSA Certified Level 1 v2.1",
    "developer-type": "PSA Certified - Device"
  }
}
}
```


Adding support for TEEP application

Section 4.3.1 of I-D.ietf-teep-protocol:

When an EAT is used, the following claims can be used to meet those requirements, whether these claims appear in Attestation Results, or in Evidence for the Verifier to use when generating Attestation Results of some form:

Requirement	Claim	Reference
Freshness proof	nonce	Section 4.1 of [I-D.ietf-rats-eat]
Device unique identifier	ueid	Section 4.2.1 of [I-D.ietf-rats-eat]
Vendor of the device	oemid	Section 4.2.3 of [I-D.ietf-rats-eat]
Class of the device	hardware-model	Section 4.2.4 of [I-D.ietf-rats-eat]
TEE hardware type	hardware-version	Section 4.2.5 of [I-D.ietf-rats-eat]
TEE hardware version	hardware-version	Section 4.2.5 of [I-D.ietf-rats-eat]
TEE firmware type	manifests	Section 4.2.15 of [I-D.ietf-rats-eat]
TEE firmware version	manifests	Section 4.2.15 of [I-D.ietf-rats-eat]

Table 1

Adding support for TEEP application (cont.)

```
$$ear-extension // = (  
  ear.teep.claims => ear-teep-claims  
)
```

```
ear-teep-claims = non-empty<{  
  ? eat.nonce => eat.nonce-type  
  ? eat.ueid => eat.ueid-type  
  ? eat.oemid => eat.oemid-type  
  ? eat.hardware-model => eat.hardware-model-type  
  ? eat.hardware-version => eat.hardware-version-type  
  ? eat.manifests => eat.manifests-type  

```


Example

```
{
  "eat_profile": "tag:github.com/veraison/ar4si,2022-10-17",
  "ear.status": "affirming",
  "ear.trustworthiness-vector": {
    "instance-identity": 2,
    "configuration": 2,
    "executables": 2,
    "hardware": 2
  },
  "iat": 1666529284,
  "ear.appraisal-policy-id": "https://veraison.example/policy/1/60a0068d",
  "ear.teep.claims": {
    "nonce": "80FH7byS7VjfARIq0_KLqu6B9j-F79QtV6p",
    "ueid": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyAh",
    "oemid": "Av8B",
    "hwmodel": "fJYq",
    "hwversion": ["1.2.5", 16384]
  }
}
```

Implementation

Golang package github.com/veraison/ar4si, and
CLI github.com/veraison/ar4si/arc

→ v0.0.2

→ Apache 2.0

→ pkg.go.dev/github.com/veraison/ar4si

Next Steps

- Continue with the experimentation
- Talk to adjacent communities
- Write up the proposal in an I-D
- Present progress update at IETF 116