IETF Hackathon – SCITT code hack

IETF 117
22-23 July 2023
San Francisco, California

SCITT high level promises

A scalable and flexible, decentralized architecture to enhance auditability and accountability across various existing and emerging supply chains. It achieves this goal by enforcing the following complementary security guarantees:

1.Statements made by Issuers about supply chain Artifacts must be identifiable, authentic, and non-repudiable;

2.such Statements must be registered on a secure append-only Log, so that their provenance and history can be independently and consistently audited;

3.Issuers can efficiently prove to any other party the Registration of their Signed Statements; verifying this proof ensures that the Issuer is consistent and non-equivocal when producing Signed Statements.

SCITT high level promises

A scalable and flexible, decentralized architecture to enhance auditability and accountability across various existing and emerging supply chains. It achieves this goal by enforcing the following complementary security guarantees:

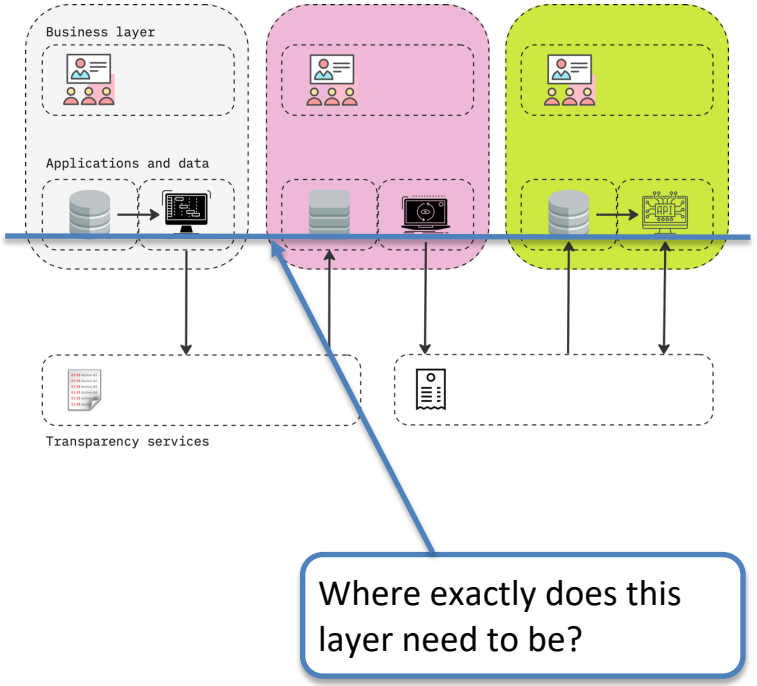
1. Statements made by Issuers about supply chain Artifacts must be identifiable, authentic, and non-repudiable;
2. such Statements must be registered on a secure append-only Log, so that their provenance and history can be independently and consistently audited;
3. Issuers can efficiently prove to any other party the Registration of their Signed Statements; verifying this proof ensures that the Issuer is consistent and non-equivocal when producing Signed Statements.

Intent of the code hack

Use the RKVST implementation of the SCITT API and COSE Merkle tree receipts to show the practicality of building systems on top of the SCITT building blocks that solve the problems the SCITT WG set out to solve.

Planned activity:

- Continue building on the open interop client
- Add initial creation of signed claims to the emulator client (not in a very secure way, but just to illustrate the steps required for a generalized solution)
- Experimenting with various application layer payloads, but particularly the Vendor Response Form
- Show different models for storage and retrieval of payloads and receipts.

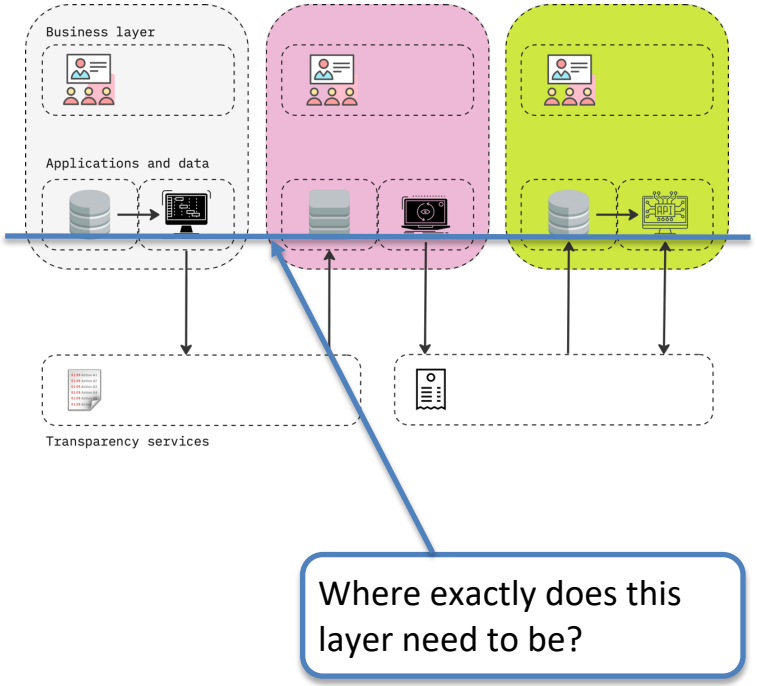


Outcome of the code hack

Use the RKVST implementation of the SCITT API and COSE Merkle tree receipts to show the practicality of building systems on top of the SCITT building blocks that solve the problems the SCITT WG set out to solve.

Hoped-for outcomes:

- Successful end-to-end demonstration of attesting and verifying a Vendor Response Form
- Driving the spec forward in understanding the different places where company identifiers might show up
- Driving the spec forward in understanding which higher level concepts (especially storage, ‘indexing’ and ‘searching’ of stuff) need to be brought into scope, and which stay up in some unspecified application layer.

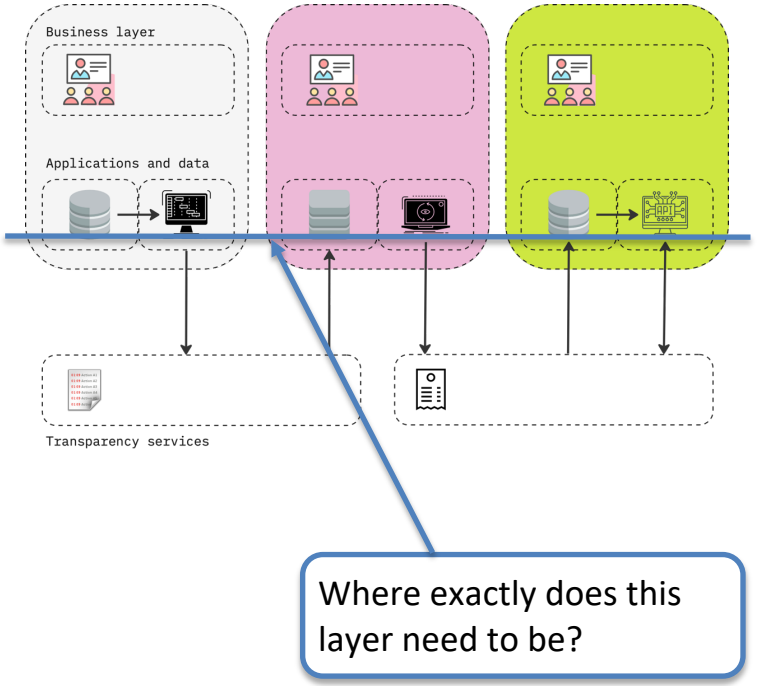


Outcome of the code hack

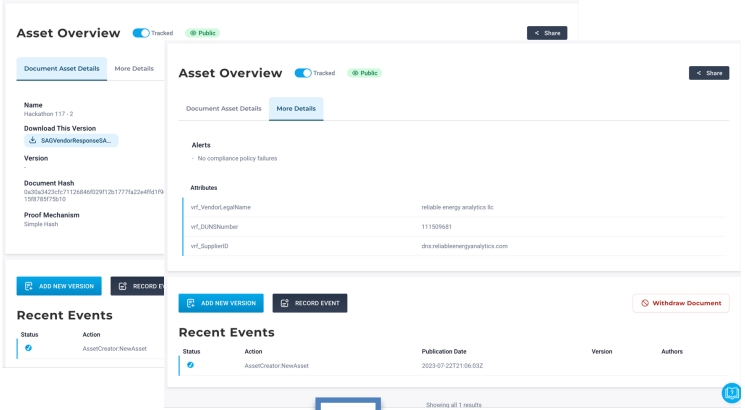
Use the RKVST implementation of the SCITT API and COSE Merkle tree receipts to show the practicality of building systems on top of the SCITT building blocks that solve the problems the SCITT WG set out to solve.

Hoped-for outcomes:

- ✅ Successful end-to-end demonstration of attesting and verifying a Vendor Response Form
- ✅ Driving the spec forward in understanding the different places where company identifiers might show up
- ✅ Driving the spec forward in understanding which higher level concepts (especially storage, 'indexing' and 'searching' of stuff) need to be brought into scope, and which stay up in some unspecified application layer.



Outcome of the code hack



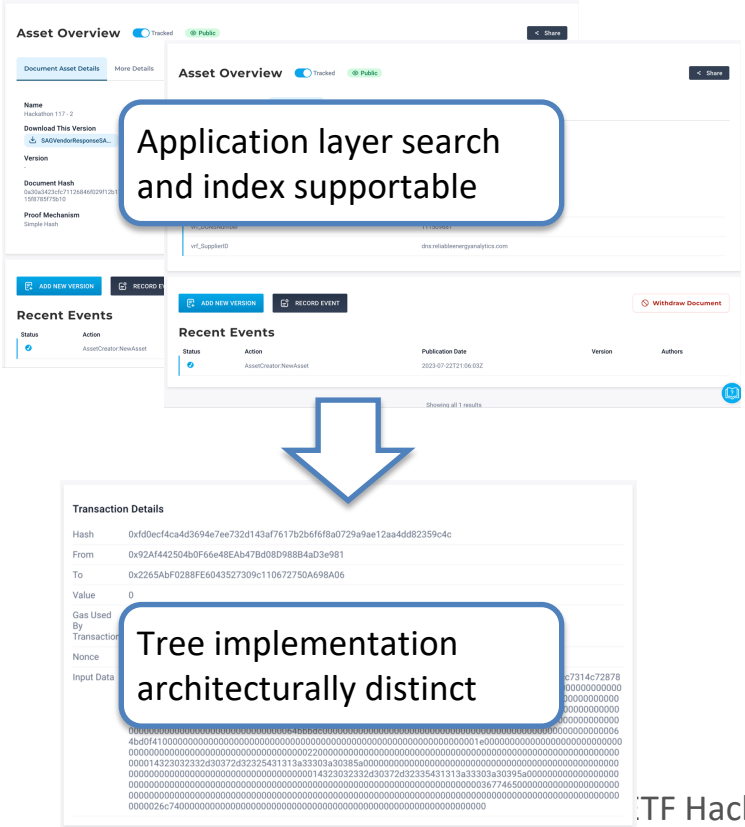

```
$ ./scitt-emulator.sh client retrieve-claim --entry-id
0xfd0ecf4ca4d3694e7ee732d143af7617b2b6f6f8a0729a9ae12aa4dd82359c4c --
out 117-vrf
Claim written to 117-vrf

$ cat 117-vrf
ˆX&?&papplication/json?mapp.rkvst.com?Xe{"identity":"","transaction_id"
:"0xfd0ecf4ca4d3694e7ee732d143af7617b2b6f6f8a0729a9ae12aa4dd82359c4c"}X
@{'6"K?????|??Y?b???)/?%?~?i???|?`??
                                                    ?????*T<bX?|(X??
```

```
$ ./scitt-emulator.sh client submit-claim --claim 117-vrf --out 117-
vrf.cbor
Claim registered with entry ID 1
Receipt written to 117-vrf.cbor

$ head 117-vrf.cbor
?X0?jservice_ideRKVSThtree_algeRKVSTiissued_atd?f?Y?o?XI?htree_algvEIP1
186NamedSlotProofsiissued_atd?f?jservice_idmapp.rkvsl|??ϙ)0?IY??{"appli
cation_parameters":{"account":"0x2265AbF0288FE6043527309c110672750A698A
06","named_proofs":[{"id":"eip1186sp:2:fv","name":"simplehash","proof":
{"nonce":"0x1","balance":"0x0","storageHash":"0xe587a9a6256dbfa59647562
97d04210b85172a1006792c08c5a4e72291463aa1","codeHash":"0xc184de5a0727ce
5f446ea9e9706b89c8bf9999d8c6072fdb2eaf12be680a318e","storageProof":[{"k
ey":"0xf80d2fd7b414f68b69ec27b5602f9b96e73c2bcedd8e4e562a27d8a9c3d4dd3a
","proof":["0xf90211a0403c6c2345aabaade194a00c1c7d19280c7301e79b7358ab6
bba0657fb337c76a0d4c41dd27e47efccc2f56d04e86d6e8ee3bb0efe489076a539a206
38b4404227a0b755f641b7064a5f580dd63cbb0606b099ce2ae6ef16ef90dc0eec25912
7f650a0b084301e717a49fc5c2957f70d537e5973a2cc239e2a1fa8ee02641dd1b80e23
a060c48c05feccde51bb
```

Outcome of the code hack




```
$ ./scitt-emulator.sh client retrieve-claim --entry-id
0xfd0ecf4ca4d3694e7ee732d143af7617b2b6f6f8a0729a9ae12aa4dd82359c4c --
out 117-vrf
Claim written to 1

$ cat 117-vrf
^X&?&papplication/
:"0xfd0ecf4ca4d369
@{'6"K?????|??Y?b??
","transaction_id"
12aa4dd82359c4c"}X
?????*T<bX?|(X??
```

Accessible through interoperable client with COSE...

```
$ ./scitt-emulator.sh client submit-claim --claim 117-vrf --out 117-
vrf.cbor
Claim registered with entry ID 1
Receipt written to 117-vrf.cbor

$ head 117-vrf.cbor
?X0?jservice_ideRKVSThtree_algeRKVSTiissued_atd?f?Y?o?XI?htree_algvEIP1
186NamedSlotProofsiissued_atd?f?jservice_idmapp.rkvsl|??ϙ)0?IY??{"appli
cation_parameters"
309c110672750A698A
06","named_proofs"
mplehash","proof":
{"nonce":"0x1","ba
9a6256dbfa59647562
97d04210b85172a100
:"0xc184de5a0727ce
5f446ea9e9706b89c8
storageProof":[{"k
ey":"0xf80d2fd7b414f68b69ec27b5602f9b96e73c2bcedd8e4e562a27d8a9c3d4dd3a
","proof":["0xf90211a0403c6c2345aabaade194a00c1c7d19280c7301e79b7358ab6
bba0657fb337c76a0d4c41dd27e47efccc2f56d04e86d6e8ee3bb0efe489076a539a206
38b4404227a0b755f641b7064a5f580dd63cbb0606b099ce2ae6ef16ef90dc0eec25912
7f650a0b084301e717a49fc5c2957f70d537e5973a2cc239e2a1fa8ee02641dd1b80e23
a060c48c05feccde51bb
```

...and CBOR structures as defined in the spec

Outcomes of the code hack

POSITIVES

- VRF use case end-to-end proven 🎉
- Other very different use cases discussed and also seem to be supported without much fuss
- Core fundamentals continue to be strong
- Highlighted (relatively) simple next step in terms of Feed specification

CHALLENGES

- Not nearly as much time to hack as I'd hoped – didn't get interoperable submission working, because...
- ...Feeds are a moving target (but at least they ARE a target 🙂)
- Client/emulator risks rotting: consider significant maintenance and quality updates alongside new API spec doc