BPSEc Updates
Interop Security Context
Security Context Template

IETF-108

Edward Birrane
Edward.Birrane@jhuapl.edu
443-778-7423
Summary
- Genart – Editorial comments, all resolved
- IANA – No issues.

Yes
- M. Westerlund

No objection
- D. Brungard, A. Cooper, R. Danyliw, B. Leiba, A. Retana, E. Vyncke
- Comments from above addressed in -22.

Discuss
- M. Kuhlewind
- B. Kaduk
- Most items believed addressed in -22. Waiting for confirmation.
BPSec Open Question #1

Should there be one security context that is considered “Mandatory to Implement” (MTI) for all BPSec implementations?

- BPSec-22 does not mandate a security context.
- States that a network/deployment must mandate a security context.
- Provides a default to be used if not other context is mandated. (Section 9.1)

Implementations of BPSec MUST support the mandated security contexts of the networks in which they are applied. If no set of security contexts is mandated for a given network, then the BPSec implementation MUST, at a minimum, implement the security context defined in [I-D.ietf-dtn-bpsec-interop-sc]. If a node serves as a gateway amongst two or more networks, the BPSec implementation at that node MUST support the union of security contexts mandated in those networks.

Recommend no change – but ADs may feel strongly.

Does not force one network to support a context it will never use.
Can BPSec be standardized absent a key exchange protocol?

- BPSec-22 does not mandate a key exchange protocol
  - Different security contexts will use different key exchange protocol
  - Some will be pre-placed symmetric (KeK or other)
  - Some will be IKE
  - Some may be DTKA.

- Key Management is not part of the normative BPSec Spec (Section 6)

There exist a myriad of ways to establish, communicate, and otherwise manage key information in a DTN. Certain DTN deployments might follow established protocols for key management whereas other DTN deployments might require new and novel approaches. BPSec assumes that key management is handled as a separate part of network management and this specification neither defines nor requires a specific key management strategy.
Consider allowing nested signatures

- BPSec-22 does not allow multiple signatures on same target
  - Proposed change: Allow multiple nodes to sign blocks
    - Nodes 1, 2, 3 independently sign a target block. (BIB1, BIB2, BIB3)
  - Security acceptors determine which BIB to pay attention to
    - Acceptor 1 may only pay attention to Node 1 signatures.
    - Acceptor 2 may only pay attention to Node 2 signatures, and so on.

- Thoughts
  - Pushes complexity into node policy configuration.
  - Why trust integrity from a non-block source? If not signed from source, it may have changed.
  - Is “intermediate” integrity a significant need?
  - Can get this with other mechanisms: new security block type or encapsulation.

Recommend no change.
Consider signature or encryption over multiple blocks.

- BPSec-22 does not allow calculating a single signature over > 1 target block
  - Example: calculate a single signature over primary block and payload block

Thoughts

- BPSec BIB and BCB are meant to be “single-target” services
  - May generate multiple security results for a single target (based on context)
  - But always a 1-many relationship: target to results.
- A multi-target service may be useful, but not part of baseline BPSec
  - BPSec provides guidelines for other security blocks (Section 10)
  - A multi-target block (many-to-many) should be defined in a different document (and only if needed).
- Proposed clarifying text change:
  - Note that BIB and BCB provides “single-target integrity” and “single-target confidentiality”

Recommend clarifying text change to BPSec-22.
Bundle Protocol Reason Codes

- A BP Node may discard a bundle for security reasons.
- Should BPSec define BP reason codes for admin records reflecting this?
  - **Reason Codes:**
    - **Missing Security Service:** Required service not present in bundle at waypoint or acceptor.
    - **Unknown Security Service:** Unknown context, parameter, etc… at waypoint/acceptor.
    - **Unexpected Security Service:** More security in bundle than expected.
    - **Failed Security Service:** Failed to verify integrity or decrypt a service at waypoint or acceptor.
    - **Conflicting Security Service:** Security blocks violate BPSec rules.

- **Thoughts**
  - *We can place them in BPSec, or in another document*
  - *Ex: Security Context Template.*
  - [https://tools.ietf.org/html/draft-birrane-dtn-scot-00#section-2.3.1](https://tools.ietf.org/html/draft-birrane-dtn-scot-00#section-2.3.1)

No recommendation
Should BPSec encode security context parms as a CBOR Map

There may be efficiencies using Map instead of Array

> Security Context Parameters (Optional):
[B. Kaduk] Why do we use an array of (index, value) tuples instead of a CBOR map?

[E. Birrane] There was no strong preference for encoding representation. Does a CBOR map result in a smaller size?

[B. Kaduk] I am not 100% sure but I think there would be some encoding efficiency from not needing repeated array framing. (Maps also help when you can assign short integer map keys to attributes that otherwise would have longer, e.g., string, names, but the Ids here are already integers so that's a > no-op.)

[E. Birrane] Recommend no change here.

Recommend no change.
Should BPSec force integrity of non-block-type-specific data?

- Protect integrity of security context parms, etc…
- Associate security block with primary block.
  - Carry signature of primary block in each security block

Thoughts

- Security results MUST include some protection of the important parts of the security block.
  - How this is done is a matter of the security contexts themselves and should not be mandated in the BPSec itself.
- Multiple ways to protect this information.
  - One approach: sign this information and carry the signature.
  - Another approach: sign each parameter (nodes can recover use defaults for corrupt parms)
  - Convey parameters as a single BLOB in the block exchanged between nodes.

Add non-normative text to “Security Context Considerations”
BPSec Open Question #8

- Should BPSec reserve some security context parm/result ids to promote commonality?
  - Create 2 registries: Security Context Parameter IDs, Security Context Result IDs
    - Specify 0-15 as “reserved” for each.
    - Specify > 16 as “defined in relevant security context document”.
    - IDs 0-15 would be shared across all security contexts.
    - IDs > 16 would be different for different security contexts.
  - Thoughts
    - May help develop commonality in security context specifications.
    - Reduce duplication of same values across multiple security contexts
    - Reduce confusion:
      - Initialization Vector is ID #1 in security context 1, but ID #17 in security context 2?

Recommend BPSec define SC Parm/Result with reserved IDs.
BPSC Interop Security Context

- No identified changes to this
  - BIB-IOP-HMAC256-SHA256
Recall what a security context is
- Cipher suite(s)
- Configuration
- Usage

Guidelines for writing security contexts
- Policy considerations
- Canonicalization considerations
- Usage, configuration, error handling

Define common standards/Enums
- Common parameters
- Common result types
- Common reason codes (BPv7)

Great value in non-normative guidance. To include suggested “table of contents” for security contexts, and critical information. Expansion of the “security context considerations” from BPSec.

Normative information relating to enumerations, states, etc... to build compatible contexts.
BPSeq Policy: Lifecycle of a Security Operation

Security Source

- Lookup (Sender Rule at Source)
- Resource Configuration Failure
- Populate
  - source_for_sop
  - sop_missconfigured_at_source
  - sop_added_at_source

Security Verifier

- Lookup (Receiver Rule at Verifier)
- Locate (Failure)
- Configuration Check Failure
  - verifer_for_sop
  - sop_missing_at_verifier
  - sop_missconfigured_at_verifier
  - sop_complied_at_verifier
  - sop_verified

Security Acceptor

- Lookup (Receiver Rule at Acceptor)
- Locate (Failure)
- Configuration Check Failure
  - acceptor_for_sop
  - sop_missing_at_acceptor
  - sop_missconfigured_at_acceptor
  - sop_complied_at_acceptor
  - sop_processed

Processing Success
- Remove
**BPSEC Policy: Events, Actions, Reasons**

<table>
<thead>
<tr>
<th>Event Mnemonic</th>
<th>Event Description</th>
<th>Actions</th>
<th>Security Reason Codes</th>
<th>BP Status Report Reason Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Lookup (Security Source)</td>
<td>sop_needed</td>
<td>Remove Sop's Tgt and Associated Sops</td>
<td>X</td>
<td>Used to set the BPCF of the security block</td>
</tr>
<tr>
<td>2. Populate</td>
<td>sop_added</td>
<td>Do Not Forward Bundle Storage</td>
<td>X</td>
<td>Follow the security block BPCF</td>
</tr>
<tr>
<td>3. Configure Resource (Failure)</td>
<td>sop_misconfigured</td>
<td>Request with this Reason Code (8-bit)</td>
<td>X</td>
<td>Follow the security block BPCF</td>
</tr>
<tr>
<td>4. Lookup (Security Verifier)</td>
<td>sop_needed</td>
<td>Remove All Sops for tgt.</td>
<td>X</td>
<td>Follow the security block BPCF</td>
</tr>
<tr>
<td>5. Configure Resource (Failure)</td>
<td>sop_misconfigured</td>
<td>Mask of BPCF to process as if tgt block could not have been processed</td>
<td>X</td>
<td>Follow the security block BPCF</td>
</tr>
<tr>
<td>6. Locate (Failure)</td>
<td>sop_misconfigured</td>
<td>Mask of BPCF for security block: Used to set the BPCF when the security block is created, or indicate the need to process the security block's current BPCF.</td>
<td>X</td>
<td>Follow the security block BPCF</td>
</tr>
<tr>
<td>7. Check Configuration (Failure)</td>
<td>sop_misconfigured</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>8. Verify (Success)</td>
<td>sop_processed</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>9. Policy Decision (Verification Failure)</td>
<td>sop_computed</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>10. Lookup (Security Acceptor)</td>
<td>sop_needed</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>11. Locate (Failure)</td>
<td>sop_misconfigured</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>12. Configure Resource (Failure)</td>
<td>sop_misconfigured</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>13. Check Configuration (Failure)</td>
<td>sop_misconfigured</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
<tr>
<td>14. Accept (Failure)</td>
<td>sop_computed</td>
<td>X</td>
<td>- Follow the security block BPCF</td>
<td></td>
</tr>
</tbody>
</table>

**14 notable events in the lifecycle of a security operation**

**Small set of actions that can be codified per event.**

**Also override some security and target processing flags.**

**Security Reason Codes**
- Missing
- Unknown
- Unexpected
- Failed
- Conflicting

**BP Status Report Reason Codes**