Key Amnesia code review for TRV1.5

DE20160629

Brief

To review and document the semantics and security code, and proposed fixes and continuing tests, for:

1. The key handling path.
2. The secure message generation path.

This is particularly with reference to the ‘key amnesia’ problem that has been experienced in TRV1/DORM1.

Security System Objectives

Current

1. Securely transmit sensitive data elements using strong encryption.
2. Authenticate received frames to prevent 3rd parties controlling the heating system or spoofing stats, etc.
3. Retain (secret) keys securely to support the crypto mechanism: **failing** as of DORM1-1.0.4 release (and matching REV10 releases).
4. Prevent replay attacks.
5. Prevent traffic analysis being used to tell when someone is home/how the heating system is being used.
6. Be lightweight enough to be used with low power MCU with limited RAM/flash memory.
7. Be lightweight enough to be used with a variety of simple low cost radio modules.
8. Be MCU and radio agnostic (not tied to particular products).
9. To protect the AES-GCM crypto used, never allow reuse of a key/IV pair.
10. Be reliable.
11. Allow device to function for at least 10 years in designed-for use cases and at expected traffic rates, eg without wearing out EEPROM.

Future

1. Guard against attacks from people with physical access to the device.
2. Make setting a new key easy, eg through pairing.

Components

The top three items (secret key storage, node associations, and AESGCM impl) are not dependent on each other in any way.

1. Secret key storage / setting / retrieval module: CLI and API front-end (OTRadioLink::OTV0P2BASE)
2. Node associations (OTRadioLink::OTV0P2BASE)
3. AES-GCM support module (OTAESGCM)
4. OTRadioLink secure frame handling including message counter (OTRadioLink::OTRadioLink) [currently msg counter interlock can clear keys]
5. Application level use (V0p2_Main + eg COHEAT)

Possible Fixes

Issue:

- see Appendix 1.

Proposed fix:

- Make sure the message counter is initialised to a unique/new (non-zero) value as part of key setting.
- Secure frame initialisation function
- Do not clear key on restart of counter.

Analysis

- Note: maybe should not reset message counter when setting a new new key in following cases (at risk of moral hazard encouraging people to reset message counters trivially, when a device hits the max counter value):
    - Key was clear before
    - New key same as old
- Issue: the relative dependencies between the app-level semantics, key handling code and AES-GCM code and OTRadioLink protocol code adds complication.
- Must not be possible to reset the message counter for an existing key
- Need an interlock between key and message counter.
- Need not to force the loading of code that particular impls do not need.
- **Worry: device has been working for 10 years, but msg counter now at max value. Temptation is to leave the key (because it needs changing in servers and elsewhere) and just reset the msg counter somehow, but that invalidates security of all previous and future TXes in principle, so want to make that v hard to do.**

Suggested process to eval/apply fix:

1. Create a version of the code with bad clearing of primary building key removed (and comments and header fixed to match)bool SimpleSecureFrame32or0BodyTXV0p2::resetRaw3BytePersistentTXRestartCounterInEEPROM(bool).
2. Review various use cases, such as recycling REV7 valves that nothing unexpected breaks.:
    a. Examine the code
    b. Test code on REV7 and REV10.
    c. There will be other cypher suites an equipment combos and key types in future.
3. Note the current application-level fix to allow rewinding the msg counter ONLY when clearing the key, to try to discourage mis-use, using OTV0P2BASE::CLI::SetSecretKey(OTRadioLink::SimpleSecureFrame32or0BodyTXV0p2::resetRaw3BytePersistentTXRestartCounterCond), and see if that or an equivalent can be pushed down below app level so that it does not keep being re-invented wrongly by each app developer.
4. Examine/enumerate what other holes there may be!

Some common use cases:

1. Setting a newly programmed REV10 and REV7s with shared key and associations.
    a. The affected code was only called on initialising Tx message counter and on key clearing. Only the first case is relevant here and will have no strange effects. The Tx counter is only used by the secure frame and will not affect anything else.
    b. Tested as far as possible without a SIM/long term test. Key is now retained across multiple resets and Txs on a REV7/REV10 pair.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected.
2. Recycling REV10s and REV7s for re-use without clearing keys, resetting IDs or not.
    a. The affected code path is never executed. The Tx counter is only used by the secure frame and will not affect anything else. Resetting the node ID is completely independent of the secure code and will not interact.
    b. Not explicitly tested but should not be affected.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected.
3. Being able to erase keys before reusing h/w, resetting IDs or not.
    a. The message counter is erased and then executes the fixed code path. Erasing the new key is avoided. The Tx counter is only used by the secure frame and will not affect anything else. Resetting the node ID is completely independent of the secure code and will not interact.
    b. Not explicitly tested but should not be affected.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected.
4. Allowing a new hub design to have more than one key.
    a. The code path is only used in stats Tx.
    b. Not implemented.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected.
5. Allowing stats hubs and boiler hubs.
    a. The code paths for handling associated nodes are independent. The issue with the REV10s was due to using the same key to send it’s own stats.
    b. Stats hub not implemented yet. Boiler hub was only affected as it uses same key for Tx of own stats.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected.
6. Allowing reverse traffic to hubs piggybacked on the TX counter just used.
    a. Not implemented yet.
    b. Not implemented yet.
    c. This is specific to counter mode authentication. Non-counter based cryptography methods should not be affected. Additionally, the

Holes Introduced:
There is no protection against within the secure frame library against reusing key/IV pairs. The app programmer has to implement it.

Other Code Paths to Check

1. Is message counter initialised before or after secure frame generator gets value? - initialised before.
2.

Appendix 1: Key loss mechanism

Link to the wiki page:
https://github.com/opentrv/OTWiki/wiki/Key-Amnesia-Investigation
Link to the investigation report:
https://github.com/opentrv/OTRadioLink/blob/master/dev/v0p2_key_amnesia/2_key_loss_isolation/key_loss_isolation_tests.md

The Issue

- Key lost on freshly programmed V0p2 devices using securable frame.
- Only on first Tx.

How To Reproduce

1. Load REV7.1.0.4.V0p2_Main.cpp.standard.hex firmware onto REV7 using USBTinyISP.
2. Set a key.
3. Wait for next Tx. The first one will succeed, the rest will fail.
4. Process can be sped up by resetting after setting the key.

The Cause

- Message counter is only initialised while the first encrypted frame is being constructed.
- If EEPROM location of the Tx message counter is all 0xFFs (as it will be after an EEPROM wipe), the key is erased as part of the initialisation process.
- As the first Tx can only happen after the key is set, one Tx succeeds:
    - The key is copied into RAM.
    - The Tx Message counter is initialised, wiping the key from EEPROM.
    - The frame is successfully transmitted.