

Hackathon on Lightweight IoT Security

Paris, France | 21-22 May 2024

parishackathon.lakewg.org



Inria



OpenSwarm

Crypto agility in lakers



Main topic:

- Planning at issues [277](#) and [278](#)
- Discussion on a common rust embedded cryptographic abstraction layer with the RIOT-rs team

Other topics:

- Interop testing with Marco Tiloca's implementation
- Fix credential-by-value in issue [273](#)
- Lots of good discussions

Porting lakers to Single-Chip μ icro Motes (SC μ M)

Keil uVision5 is used to obtain the compiled binary firmware on SC μ M



Synthesis: lakers-c can be suitable, but:

- Software-based crypto backend
 - Required flash memory size > SC μ M instruction SRAM size (64kB) ❌
- Hardware-based crypto backend
 - SC μ M-3C has no hardware accelerators ❌

Solution: use nRF52840-DK as crypto backend and the radio on SC μ M (temporary), or just wait to get the next version of SC μ M having hardware accelerators ✅

Practical work:

- getting started with lakers (had to debug lots of errors to build it on my Windows machine)
- learning about the IEEE802.15.4 compliant radio on SC μ M

RIOT-rs: Integrating CoRE security

- Participated successfully in EDHOC interop.
- Demonstrated RIOT-rs as a platform for protected CoAP operations.
- Coordination for development with Lakers.



Interop testing of EDHOC

- Marco (Eclipse Californium, I/R), Christian (Lakers on RIOT-rs, I/R)
 - Ciphersuite 2; Method 3; CCS credentials referred by 'kid'
 - Successful interop - First time using the EDHOC + OSCORE combined request [1]
 - Found things to fix and now fixed!
 - Christian: encoding of credentials transferred by value
 - Marco: local retrieval of OSCORE Security Context when using the combined request
- Marco (Eclipse Californium, R), Geovane (Lakers, I)
 - Ciphersuite 2; Method 3; CCS credentials referred by 'kid'
 - Successful interop
 - Found things to fix:
 - Geovane: handling received connection identifier encoded as a CBOR byte string
- Stefan with Marco or Christian (started)

[1] <https://datatracker.ietf.org/doc/draft-ietf-core-oscore-edhoc/>

EDHOC and OSCORE profile of the ACE framework

- Had to prioritize EDHOC interop and related side meetings
- No big outcome: just set up the development ground at
 - <https://bitbucket.org/marco-tiloca-sics/ace-java/src/edhoc-oscore-profile/>
- Confirmed next steps
 - Set a node to be both ACE Client and EDHOC Initiator
 - Set a node to be both ACE Resource Server and EDHOC Responder
 - Morph the OSCORE profile into the EDHOC and OSCORE profile

Interop testing of EAP-EDHOC

Developing & Testing EAP-EDHOC implementations
UM and UNIOVI's (so far)

UM implementation

- *uoscore-uedhoc implementation (3.0.2)*
- EAP peer: wpa_supplicant 2.10 or 2.11-dev (also supports EAP-TLS (v1.3))
- EAP authenticator: hostapd 2.10 or 2.11-dev
- EAP server: Freeradius 3.2.3

UNIOVI implementation

- *uoscore-uedhoc (1.0.5)*
- EAP peer and authenticator: OpenPANA 0.2.4
- EAP server: FreeRADIUS 3.2.1

Interop plan:

- **UM's EAP peer and authenticator (wpa_supplicant/hostapd) and UNIOVI's EAP server (FreeRADIUS)**
 - Different EAP method type (fixed)**
 - MSK derivation was different (fixed)**
 - uoscore-uedhoc version 1.0.5 and 3.0.2 interoperate!!!**
- **UM integration of last version of uoscore-uedhoc (3.0.4) (RFC compliant)**

Future plan:

- Continue with UNIOVI's EAP peer and authenticator (OpenPANA) and UM's EAP server (FreeRADIUS)
- Integrate EAP peer in a real IoT device
- Fragmentation support

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	RADIUS	170	Access-Request id=0
2	0.001849	127.0.0.1	127.0.0.1	RADIUS	112	Access-Challenge id=0
3	0.004262	127.0.0.1	127.0.0.1	RADIUS	221	Access-Request id=1
4	0.008866	127.0.0.1	127.0.0.1	RADIUS	511	Access-Challenge id=1
5	0.017552	127.0.0.1	127.0.0.1	RADIUS	559	Access-Request id=2
6	0.023342	127.0.0.1	127.0.0.1	RADIUS	120	Access-Challenge id=2
7	0.063484	127.0.0.1	127.0.0.1	RADIUS	193	Access-Request id=3
8	0.066048	127.0.0.1	127.0.0.1	RADIUS	278	Access-Accept id=3

Frame 8: 278 bytes on wire (2224 bits), 278 bytes captured (2224 bits)	
Linux cooked capture v2	
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	
User Datagram Protocol, Src Port: 1812, Dst Port: 50500	
RADIUS Protocol	
Code: Access-Accept (2)	
Packet identifier: 0x3 (3)	
Length: 230	
Authenticator: 7ec146a9cdbc97b903cec5929e522026	
[This is a response to a request in frame 7]	
[Time from request: 0.002564000 seconds]	
Attribute Value Pairs	
AVP: t=Vendor-Specific(26) l=90 vnd=Microsoft(311)	
AVP: t=Vendor-Specific(26) l=90 vnd=Microsoft(311)	
AVP: t=EAP-Message(79) l=6 Last Segment(1)	
Type: 79	
Length: 6	
EAP Fragment: 03030001	
Extensible Authentication Protocol	
Code: Success (3)	
Id: 163	
Length: 4	



Formal Verification of Attested TLS

- Good participation in side meeting (9 attendees in total)
 - 2x one-hour sessions
- Several discussions and feedback (mostly ending up in philosophical ends)
- Still find no formal justification of `derive_secret` in TLS key hierarchy
 - John Mattsson also agreed that there is no practical issue by missing this
- Still find no other solutions to fix RA-TLS other than making it stateful within pre-handshake attestation
- Discussed formal semantics with Benjamin Lion. Thanks!
- Proposed some directions for “Attestation over EDHOC” draft
 - Will follow up on the ML

Constrained BRSKI (cBRSKI) Onboarding for Thread devices

Progress: added function to OT-NS simulator to transport UDP datagrams between a Thread Border Router and an external server (e.g. BRSKI Registrar); and IP packets between any mesh node and external server.

- Basic unit tests added
 - Includes Python aiocoap external server as part of the tests
- Lots of debugging & code fixing!
- Next step: add cBRSKI specific handling in the Border Router.

Hackathon on Lightweight IoT Security

Paris, France | 21-22 May 2024

parishackathon.lakewg.org



Inria



OpenSwarm