

Secure communication with own embedded applications

Getting started with Ariel OS, CoAP and trust anchors

Christian Amsüss

chrysn@fsfe.org

2025-08-15

FrOSCon, Embedded Systems Devroom

Plan for today

1. Introduction and terminology
2. Setting up an application with Ariel OS
3. Communication and security components by example
4. Security establishment outlook

Scope

embedded devices

Scope

IoT devices

constrained devices¹

¹RFC 7228

Scope

Ballpark

16–100KiB RAM

64KiB–1MiB flash

Scope

Ballpark

16–100KiB RAM

64KiB–1MiB flash

WiFi, 6LoWPAN, LoRaWAN

Tools

HW STM32WB55 kit

OS Ariel OS

net USB Ethernet

app CoAP

sec EDHOC

Tools

HW STM32WB55 kit

OS Ariel OS

net USB Ethernet

app CoAP

sec EDHOC

Raspberry Pi

GNU/Linux

WiFi

HTTP

TLS

Writing an application

Copy this build it at home²

```
$ cargo generate --name my-coap-server --git \  
  https://github.com/ariel-os/ariel-os-template  
$ cd my-coap-server  
$ laze build -b st-nucleo-wb55 run  
[...]  
[INFO ] Hello from main()! Running on a st-nucleo-wb55 board.
```

²Details on https://christian.amsuess.com/blog/website/2025-03-27_ariel_coap/, based on Ariel OS 0.1; full code on <https://codeberg.org/chrysn/ariel-coap-tutorial-example>.

Writing a CoAP server

into `src/main.rs`

```
#[ariel_os::task(autostart)]
fn main() {
    use coap_handler_implementations::{HandlerBuilder,
        SimpleRendered, new_dispatcher};

    let handler = new_dispatcher()
        .at(&[], SimpleRendered("Hello from my CoAP server"))
        .at(
            &["private", "code"],
            SimpleRendered("The password is 'mellon'."),
        );

    ariel_os::coap::coap_run(handler).await;
}
```

Writing a CoAP server

Of course it can't be *that* easy

Add into *laze-project.yml*:

```
apps:  
  - name: my-coap-server  
+   selects:  
+     - coap-server  
+     - ?coap-server-config-storage
```

Add into *Cargo.toml*:

```
[dependencies]  
ariel-os = { path = "build/imports/ariel-os/src/ariel-os", features  
ariel-os-boards = { path = "build/imports/ariel-os/src/ariel-os-boar  
+  
+coap-handler-implementations = "0.5.4"
```

Writing a CoAP server

Security policy from the start

New file *peers.yml*:

```
- from: unauthenticated
  scope:
    /.well-known/core: GET
    /: GET
```

Demo time: Starting the server

With apologies for the 1990s' address family.

```
$ laze build -b st-nucleo-wb55 run
[... ]
[INFO ] IPv4: UP
[INFO ]   IP address:      10.42.0.175/24
[INFO ]   Default gateway: Some(10.42.0.1)
[INFO ] IPv6: DOWN
[INFO ] Starting up CoAP server
[INFO ] CoAP server identity: {8:{1:{1:2, 2:h'', -1:1, -2:h'bc19b400a
[INFO ] Server is ready.
```

Demo time: Using a client

```
$ aiocoap-client coap://10.42.0.175/.well-known/core
# application/link-format content was re-formatted
<>,
</private/code>
$ aiocoap-client coap://10.42.0.175/
Hello from my CoAP server
$ aiocoap-client coap://10.42.0.175/private/code
4.01 Unauthorized
```

What we have

communication established

sensitive resources inaccessible

Client-side setup

This is not a *pretty* aspect of aiocoap, so far.

Add opportunistic encryption by creating *client.diag*:

```
{
  "coap://10.42.0.175/*": {
    "edhoc-oscore": {
      "suite": 2,
      "method": 3,
      "own_cred": {"unauthenticated": true},
      "peer_cred": {"unauthenticated": true},
    }
  },
}
```

```
$ aiocoap-client --credentials client.diag coap://10.42.0.175/
Hello from my CoAP server
```

What we have

passive attackers defeated

Whom are we talking to?

The server told us:

```
$ laze build -b st-nucleo-wb55 run
```

```
[...]
```

```
[INFO ] CoAP server identity: {8:{1:{1:2, 2:h'', -1:1, -2:h'bc19b400a
```

Change in *client.diag*:

```
-   "own_cred": {"unauthenticated": true},  
+   "peer_cred": {"unauthenticated": true},  
+   "peer_cred": {14: {8: {1: {1:2, 2:h'', -1:1, -2:h'bc19b400aba518  
    }
```

What we have

server authenticated

Who are we?

```
$ aiocoap-keygen generate myself.cosekey --kid 01  
{14:{8:{1:{1:2,2:h'01',-1:1,-2:h'4f3c3330eb7ed2c1c1c39d5814476184c972
```

Change in *client.diag*:

```
-   "own_cred": {"unauthenticated": true},  
+   "own_cred_style": "by-key-id",  
+   "own_cred": {14:{8:{1:{1:2,2:h'01',-1:1,-2:h'decf4b5f1148b2186  
+   "private_key_file": "myself.cosekey",  
   }
```

What may we do?

Add in *peers.yml*:

```
- kccs: |
    {8:{1:{1:2,2:h'01',-1:1,-2:h'decf4b5f1148b2186fea6461b21d08592333
scope: allow-all
```

```
$ aiocoap-client --credentials client.diag coap://10.42.0.175/private
The password is 'mellon'.
```

What we have

mutual authentication

See https://christian.amsuess.com/blog/website/2025-03-27_ariel_coap/ and <https://codeberg.org/chrysn/ariel-coap-tutorial-example> for a more elaborate example with finer grained access control.

What we have

copying public keys like it's
.ssh/authorized_keys

Outlook

...like it's `.ssh/known_hosts`?

```
$ aiocoap-client 'coap://[2001:db8::1]/.well-known/core'
```

```
The authenticity of host '2001:db8::1' can't be established.
```

```
COSE credential is {8:{1:{1:2,-1:1,-2:h'bc19b400aba518eeaece9c01c7e03
```

```
Are you sure you want to continue connecting (yes/no/[credential])?
```

Outlook

...like it's TLS?

client:

```
"peer_cred": {"signed_by": "WebPKI"},
```

Outlook

...like it's DANE?

client:

```
"peer_cred": {"refer_to": "dnssec"},
```

Outlook

...like it's DANE?

client:

```
"peer_cred": {"refer_to": "dnssec"},
```

TLSA / SSHFP

Outlook

...like it's JWTs (JSON Web Tokens)?

server:

```
- cwt: |  
  {1:2,-1:1,-2:h'71469646966b8c3be707f82113f922bd79709dc  
  scope: as-in-cwt  
  aud: my-device-name
```

client:

```
"own_cred": {13:<<18([<<{1:-7}>>,{4:h'4173796D6D65747269  
"private_key_file": "for-cbor-web-tokens.cosekey",
```

Outlook

...like it's OAuth³ (i. e., there is a trusted Authentication Server)?

client:

```
"both_cred": {  
  "as-url": "https://iam.example.com",  
  "aud": "my-device-name",  
}
```

³Here called ACE, Authentication and Authorization for Constrained Environments, [RFC 9200](#)

Outlook

...like it's OAuth? Works⁴.

oscore.gitlab.io/coap-ace-poc-webapp/

CoAP ACE PoC: The App

Devices

Search nearby devices

- Device currently not connected.

Security association: d00 at https://keycloak.co

Token: encrypted for EDHOC profile (ca1bd773)

OSCORE: not established

► Request token manually

Logins

- Logged in as Junior Technician at https://keycloak.co

Internet connection available force offline mode

your account

Sign In

⁴Not with Ariel OS, but its underlying library. And the client is a different story.

Outlook

...like it's OAuth – with better tools.

```
$ laze build -b st-nucleo-wb55 run
```

```
[...]
```

```
Device named st-nucleo-wb55:01234567, generated credential {1:{1:2,-1
```

```
Registering at configured AS https://iam.example.com.
```

```
Already logged in, no need to open browser.
```

```
[...]^Z
```

```
$ aiocoap-client 'coap://[2001:db8::1]/private/code'
```

```
# No credentials found; to authorize, log in via
```

```
# https://iam.example.com/?i%20cant%20write%20OAuth%20by%20hand
```

```
[...]
```

```
# Thank you. Continuing with token for st-nucleo-wb55:01234567.
```

```
The password is 'mellon'.
```

Outlook, final

Device lifecycle

1. Build
2. Sell
3. Ship
4. Run
5. ...
6. (not sure, I stopped reading at "Profit.")

Outlook, final

Device lifecycle

1. Build – generate IDevID, store at manufacturer
2. Sell – introduce to buyer AS
3. Ship
4. Run – AS gets to know device, 3-party handshake once

zero-touch provisioning

Summary

1. Ariel OS apps are easy to build.
2. Its CoAP stack provides end-to-end security.
3. Mutual authentication works, but takes some copy-pasting.
4. Specs for runtime credential generation are implemented.
5. Tools will be beaten until usability improves.
6. Zero-touch provisioning on the horizon.

Christian Amsüss
chrysn@fsfe.org