GSoC Final Report

Author: Vedang Joshi

Published: August 16, 2021 · 4 min read

The second evaluation for the Google Summer of Code 2021 has begun. This has been a wonderful experience so far. Fortunately, we have completed all the work promised in the previous blog post. Here are some quick links to the PRs, repos, and blogs related to my project.

- DRG
- Drogue IoT
- GSoC Proposal

PRs

- All commits
- #81 Added primary trust anchor support
- #86 Added --cert flag
- #87 Added ca key verification
- #89 Added RSA key generation and algo parameter
- #90 Added custom key support
- #91 Added test cases trust.rs
- RFC

Blog

- GSoC Phase 1
- GSOC, Drogue IoT & the Community

Implementation

We can authenticate our devices to Drogue IoT using either passwords or X.509 certificates. The latter method is better as it is scalable and more secure. To know more about the implementation kindly refer to the phase 1 blog.

In the past few weeks, I added support for generating RSA keys using the RSA crate. The private keys can be generated from any of the 3 algorithms - RSA, ECDSA, EdDSA. To specify an algorithm you would need to specify the `--algo` parameter.

```
drg trust create <app_name> --algo RSA
```

To save you the trouble of specifying the parameter every time, we leveraged the context functionality in drg. Thus you can specify a default signature algorithm for a context.

```
drg context set-default-algo RSA
```

Other than generating keys and certificates, users can also use existing keys. The only constraint here is it should strictly be in a PKCS8 format. To do this user can specify the `--key-input` parameter. To better understand the required format you can check out Ring's documentation.

Assuming you are using OpenSSL to generate

- RSA keys

```
openssl genrsa -out key.pem 2048
```

- ECDSA keys

```
openssl ecparam -name prime256v1 -genkey -noout -out key.pem
```

The only additional step is

```
openssl pkcs8 -topk8 -nocrypt -outform der -in key.pem > private.pk8
```

Now, you can use the `private.pk8` file to sign app or device certificates using

```
drg trust create <app_name> --key-input <key_file>
```

In the intial proposal, there were talks about creating a "key-store" to securely save all these private keys, but as we progressed the decision about where to store the keys was left to the user. That's when I thought, it would be a good idea to demonstrate how to store keys in a solution like LastPass and use it in drg. Frankly, it is just some 'terminal trickery' :).

Assuming you have LastPass cli installed, and you are logged in DRG and LastPassCli. To store app private key

```
printf "Private Key: %s\n" "$(drg trust create --app app_name)" | lpass
add --non-interactive --sync=now "app_name" --note-type=ssh-key
```

Now, to use that key to sign device cert+key

```
drg trust enroll d1 --ca-key <(lpass show app_name --field="Private Key")
--app app_name
```

In addition to this, we also tried writing unit test cases for various functions used by the `trust` subcommand. These test cases provide a ~92% code coverage in the `trust.rs` file. Required changes were also made to the ci file to run the test cases on each commit.

State of the project

The majority of functionality related to the `Trust-anchor management` has been implemented and merged to the main branch of DRG.

Challenges and learning

- The primary challenge in this project was acquiring knowledge, I was extremely new to cert-based authentication, so I learned many things on the way.
- We had to adjust by introducing a `--cert` flag, due to some device naming limitations, but we plan to overcome that by implementing alias functionality.
- RSA key generation was done using RSA crate. Initially, it was not compatible with the Ring crate we have used so far. This was an obstacle. I reported this on the RSA crate's community channel, they were already aware of this and had solved it on the main branch. Few days after this conversation, they made a new release and we were able to implement RSA key generation in our code.

The learnings I've got throughout this time are unparalleled and they will stay with me for a long time. I shall continue to make more such contributions.

Thank you

GSOC DrogueIoT IoT Github Open source