Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund

FaaS_XRA: A highly customizable remote attestation framework for Intel SGX

Project Thesis'20: TFaaS - First Draft

Joshua Heinemann, April 28, 2020

FaaS_XRA - Goals

FaaS eXtended Remote Attestation:

+ Highly customizable
  - Only measure desired features
+ offers an interface that is self-explanatory
+ Fast
+ Low resource consumption
  - CPU time as well as memory usage

FaaS_XRA - Concept

Concept:

- User enables all desired measurements
  - e.g. `XRA_CIPHER`, `XRA_WASM_CODE`

  → Highly customizable
- *Static features* will be measured during framework bootstrap
- *Dynamic features* will be measured by invocation of `xra_gen_report`

Example:

```
1  ...
2  xra_status_t status;
3  if((status = xra_init(XRA_CIPHER|XRA_WASM_CODE|XRA_CUSTOM)) != XRA_SUCCESS)
4          goto fail;
5  if((status = xra_reg_wasm_code((void*)wasm_binary, sizeof(wasm_binary)))
6   != XRA_SUCCESS)
7          goto fail;
```

FaaS_XRA - Measurable Features

Static features:

- Enclave features
- WAMR features
  - Runtime-specific features e.g. *heap-size*
  - Native functions

Dynamic features:

- WASM application code
- Cipher used for encryption of the WASM application code

FaaS_XRA - Report

Report consists of:

- Measured features
- Custom field
  - Stores user-specific content
  - e.g. Comments or Keys

Report process:

1. Measure all features
2. Generate `xra_report_t`
3. Calculate `xra_report_t`'s hash sum
4. Generate `sgx_report_t` and use the hash sum as `sgx_report_data_t`
5. Return both reports

Example:

```
1  //All features are already registered
2  ...
3  xra_status_t status;
4  sgx_status_t stat;
5  xra_report_t* xra_report;
6  sgx_status_t* sgx_report;
7  sgx_target_info_t target;
8  if((stat = sgx_self_target(&target)) !=
9     SGX_SUCCESS)
10        goto fail;
11 if((status = xra_gen_report(&target,
12    &xra_report, &sgx_report)) != XRA_SUCCESS)
13        goto fail;
14 ...
```