TEE Compile

Background

In the current technological landscape, there's a glaring absence of mechanisms to verify that an executable file has been compiled from a specific source code. This gap poses significant challenges in terms of security, transparency, and trust, as there is no definitive way to prove the authenticity of the compiled software.

To address this issue, we are introducing a method that involves standardizing the compilation process within an enclave environment. An enclave is a protected, isolated area of execution, where code can be run in confidentiality and integrity, safeguarded from potential tampering or unauthorized access.

Workflow

?

Roles:

- Action Runner:
- TEE Compile:
-
    - Responsible for compiling the source code into executable files.
-
    - Divided into Client and Worker two parts, the Client is responsible for running Enclave. Since the enclave cannot directly communicate with the outside, this part of the work is done through communication between Worker and Client.
-
    - Client will submit the source code to Worker, and the Worker is responsible for collecting relevant information, and then compiling, finally generating the Executable Files, and returning them to the Client.
-
    - Client will save the Executable Files locally.
- *
-

Initialization Work

1. The Project Owner is responsible for running an Aws Nitro VM and initializing it, including configuring it as a GitHub Action Runner and installing TEE Compile.
2. Download the corresponding language's enclave image (.eif) from a trusted source.
3.
    1. Different languages and different compiler versions will have different images.
4. 2.
5. Create a build.json in the project root directory, for example:
6. ```
7. Copy
8. {
9. "language":"rust",
10. "input":{
11. "cmd":"cargo build"
12. },
13. "output":{
14. "files":[
15. "target/debug/rust-project"
16.]
17. }
18. }
19. ```
20.

New Version Release

1. TriggerBuildTool(Client)
2. in the runner through GitHub Action for compilation.
3. The Action Runner will download the code, and execute theBuildTool(Client)
4. ,then scan the configuration files in the project, using an enclave image base on the project configuration.
5. BuildTool(Worker)
6. will obtain the code via Builder, understanding from the configuration files how to compile the project (language, compilation commands to be executed, information to be collected, and list of output files).
7. BuildTool(Worker)

8. will execute the compilation program, hijacking network traffic in a specific way
9. BuildTool(Worker)
10. will collect the produced files, calculate the hash, and generate an Attestation Report (containing information about input and output).
11. Package the Executable Files and Attestation Report into a tar file, outputting throughBuildTool(Worker)
12. to the outside world, such as GitHub, and ultimately provide to Project User.
13.

Report Verification

1. Whether the hash of the produced executable files is the same.
2. Whether the compiled image is known to be trustworthy.
3. Verify the issuer of the report.
4.

Challenges

1. The attestor can recreate the compilation environment according to the Attestation Report, thus verifying.
2. The majority of votes wins.
3.

Previous FAQ Next Getting Started Last updated14 hours ago On this page *Background * Workflow * Initialization Work * New Version Release * Report Verification * Challenges

Was this helpful?