ACM CCS 2025 Artifact Appendix: Attestable Builds: Compiling Verifiable Binaries on Untrusted Systems using Trusted Execution Environments

Daniel Hugenroth*
dh623@cam.ac.uk
University of Cambridge
Cambridge, United Kingdom

Mario Lins*
mario.lins@ins.jku.at
Johannes Kepler University
Linz, Austria

René Mayrhofer
rm@ins.jku.at
Johannes Kepler University
Linz, Austria

Alastair R. Beresford
arb33@cam.ac.uk
University of Cambridge
Cambridge, United Kingdom

A ARTIFACT APPENDIX

In our submitted paper we present Attestable Builds (A-Bs) that provides strong source-to-binary correspondence with transparency and accountability. In particular, A-Bs focuses on cloud-based CI/CD pipelines potentially provided by an untrusted provider. We implemented a prototype using AWS Nitro Enclaves as the underlying Confidential Computing technology. In addition, we evaluated the performance of our system and formally verified our new protocol. The artifacts can be used to run the performance evaluation and the formal verification.

A.1 Abstract

Our primary artifact is an implementation of Attestable Builds and a suite of evaluation scripts. We provide our code as a Zenodo-archived ZIP file and our project samples in a GitHub repository. The implementation can be run against our sample projects using AWS Nitro Enclaves to reproduce the main results of our paper. We also include our formal model and instructions on how to run them with Tamarin.

A.2 Description & Requirements

We provide a single ZIP file that includes all relevant documentation and source code. The ZIP file contains a *walkthrough.md* file located in the ‘attestable-builds-artifact-eval‘ directory with all the details to prepare the AWS machine, run the performance evaluation, and to perform the formal verification. The source code of A-Bs is located in the directory called *attestable-builds-artifact-eval*.

A.2.1 Security, privacy, and ethical concerns. Provisioning resources on AWS may incur charges that can be surprising to unfamiliar users. EC2 instances continue to incur charges even when they are “stopped”—they only stop to be considered billable resources once they are fully “terminated”. We recommend user to first familiarize themselves with the AWS pricing system.

A.2.2 How to access. The archived version of our artifacts is available at Zenodo (https://doi.org/10.5281/zenodo.17026892). In addition, we are also making our repositories available through GitHub for ease-of-testing. The *attestable-builds* repository can be found on https://github.com/lambdapioneer/attestable-builds/tree/artifact-eval and the repository containing the project samples on https://github.com/lambdapioneer/ab-samples. Note that the *attestable-builds* repositories has a dedicated branch for the artifact evaluation called *artifact-eval*.

A.2.3 Hardware dependencies. As we utilize AWS Nitro Enclaves the prototype implementation must be run on an AWS EC2 instance. For this evaluation we recommend to use the following configuration:

- Instance type: m5a.8xlarge
- Amazon Linux 2023 kernel-6.1 AMI
- 64GiB gp3 Storage
- Nitro Enlcave support enabled

A.2.4 Software dependencies. The local computer should run a modern Linux OS and an SSH client to connect to the AWS instance. Since our prototype integrates with GitHub actions, the user will also need a GitHub account. The software dependencies on the AWS instance are installed when executing our *artifact-eval-setup.sh* script located under *attestable-builds-artifact-eval/scripts*.

A.2.5 Benchmarks. The sample projects, used to demonstrate the practicability of our design and to evaluate performance metrics, are hosted on GitHub (https://github.com/lambdapioneer/ab-samples). We suggest that the user forks this repository and recreates the project-specific branches.

A.3 Set Up

We recommend to read the detailed walkthrough documentation that contains relevant details to prepare the AWS machine, run the performance evaluation, and perform the formal verification. The walkthrough can be found in the *walkthrough.md* file contained in the ZIP file or in the *walkthrough.md* in the linked GitHub repository.

A.3.1 Installation. The installation requires multiple steps to prepare the AWS instance and the GitHub repository, which contains the sample projects for running the evaluation. Download the the artifact ZIP file from Zonedo (https://doi.org/10.5281/zenodo.17026892), and follow the walkthrough documentation included in the *walkthrough.md* file.

*These authors contributed equally to this work.

A.3.2 Basic test. We provide two test suites: one suite to verify the basic setup of the machine and one suite containing various smoke tests to verify the evaluation setup. A detailed instruction for running both test suites is included in the *walkthrough.md* file. The smoketest run a subset of targets and configurations providing coverage to spot the most common problems.

A.4 Evaluation Workflow

For the evaluation, the user will set up an AWS EC2 instance using the instructions and code provided in the main artifact repository. As part of the set-up, they will also instantiate a repository with the sample builds on GitHub. Figure 1 provides an overview of our setup. Within the EC2 instance, the evaluation script will later run workflows inside the sample build repository through its integration with GitHub actions. The log files of these executions will be collected within the `evaluations/scenarios. . .` folders and are later processed using Python scripts.

A.4.1 Major claims. The paper makes the following major claims that are related to the artifacts provided:

(C1): The system builds a variety of open-source projects inside a secure setup using AWS Nitro Enclaves and an inner sandbox. See §1 and §4 of the main paper.
(C2): The system does so with a relatively small overhead of around 42s start-up time and 14% increase in build time (68% for hardened sandboxes). See §1 and §4 of the main paper. In particular, Figures 4, 5, and 6.
(C3): The provided formalized proofs work pass as expected when run with Tamarin. See list of contributions in the introduction of the main paper. See §1 and §5 of the main paper.

A.4.2 Experiments. All experiments are described in detail in a dedicated `walkthrough.md` file[1] within the repository and the ZIP archive. They contain multiple steps that are individually annotated with the expected time required in both interactive spans (human time) and when experiments run automatically (machine time). Here we report the expected overall time.

(E1): [2h human time + 20h machine time + AWS EC2 2x.large instance]: this experiment replicates the main claims C1, C2 of the practical evaluation. In particular, it reproduces the plots shown in the paper. We provide instructions to run it with a single iteration (instead of the three iterations we use in the paper) to keep the total runtime short.
Preparation: Following the walkthrough guide the user will setup an EC2 instance and a GitHub repository that are required for the main evaluation.
Execution: After the preparation, the main evaluation (which takes approximately 18 hours compute time) is executed through an evaluation script.
Results: The final analysis notebook output plots that match Figures 4, 5, and 6 of the main paper.
(E2): [10min human time + 5min machine time]: this experiment replicates the claim C3 regarding the formal proofs.
Preparation: Following the walkthrough guide the user will install Tamarin.
Execution: Using the Tamarin proof, the proofs will execute automatically.
Results: The Tamarin software will output a success message.

[1] https://github.com/lambdapioneer/attestable-builds/blob/artifact-eval/walkthrough.md

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926.

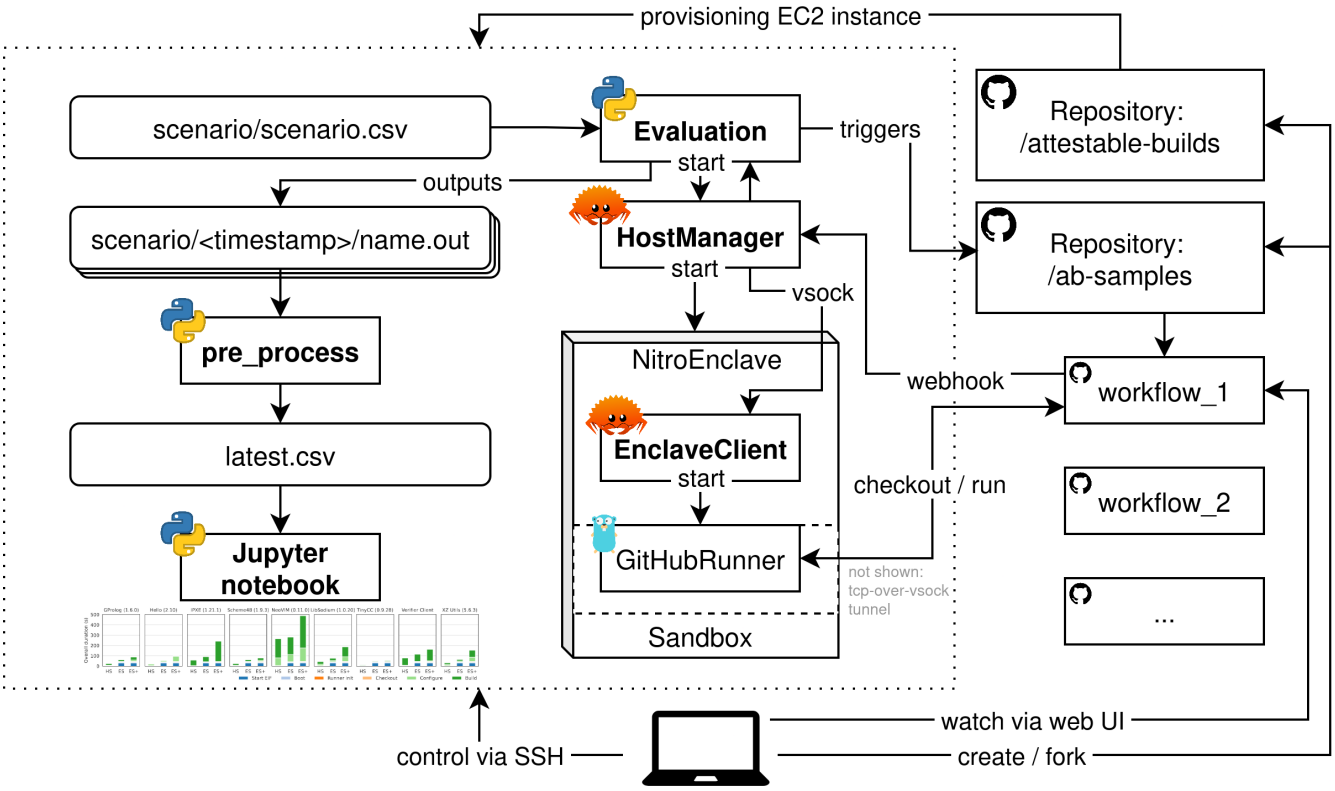

Figure 1: Overview of the evaluation setup with indication of the used programming language for the respective components. The main repository `attestable-builds` is used to provision a large EC2 instance. The `ab-samples` repository is then initialized to contain one branch per target. The evaluation scripts read the respective `scenario.csv` to learn about the intended list of targets and configurations. It then trigger workflows one-by-one inside the `ab-samples` repository and starting *HostManager* with the respective configuration (e.g. HS, ES+, …) which then starts the *EnclaveClient*. This figure shows an example that is reflective of ES/ES+, i.e. an enclave with an inner sandbox. However, in non-enclave configurations, that are included as a baseline, the *EnclaveClient* might run directly on the host. In all configurations, the enclave client starts a GitHub runner that will checkout the workflow and run the required steps of the build. Since these interact with the GitHub actions normally, they can be observed via the regular web UI. When all workflows have finished, the `pre_process` script converts the gathered log outputs into a `.csv` file. This can then processed with the provided a Jupyter notebook.