ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

A Artifact Appendix

A.1 Abstract

This artifact appendix describes the complete workflow to setup Bedrock. It includes an artifact check-list, description of hardware/software dependencies, experiment workflow, and the expected results.

A.2 Artifact check-list (meta-information)

- **Compilation:** GCC v7.5.0, Tofino SDE v8.4.0, Netronome SDE v6.1.0.
- **Binary:** Source code included to generate binaries.
- **Run-time environment:** End host codes are tested on x86 servers running 64-bit Ubuntu18.04 OS with BPF compiler collection. Both servers and switch need root access.
- **Hardware:** Intel/Barefoot Tofino1 switch ×1, x86 server with Mellanox ConnectX-4 RNICs ×4, x86 server with Netronome Agilio CX ×1.
- **Metrics:** Throughput, latency, CPU utilization, attack and defense effectiveness.
- **Output:** The server, client and attacker programs output messages indicating whether the attack succeeds. Throughput and latency can be measure by tools like `tcpdump`. CPU utilization can be measurement via tools like `top`.
- **Experiments:** See Section A.5 and Section A.5.
- **How much disk space required (approximately)?:** 1GB (dependencies not included).
- **How much time is needed to prepare workflow (approximately)?:** Compiling all programs needs about 1 hour (installation of software dependencies and hardware is not included).
- **How much time is needed to complete experiments (approximately)?:** About 2 hours to see the effect of all attacks and defenses.
- **Publicly available?:** Yes, code is available on GitHub.
- **Code licenses:** MIT license

A.3 Description

A.3.1 How to access

Bedrock is publicly available at the following GitHub repository: https://github.com/alex1230608/Bedrock. (commit: 4eef2619d7fb007b4c8ed690c6d78e8fea377455)

A.3.2 Hardware dependencies

To run Bedrock, it requires four x86 servers connecting to an Intel/Barefoot Tofino switch or a Netronome Agilio CX SmartNICs through Mellanox ConnectX-4 RNICs.

A.3.3 Software dependencies

Our experiments are performed on x86 servers running 64-bit Ubuntu 18.04, but similar Linux distributions should also work. To enable RDMA, Mellanox MLNX_OFED driver must be installed on the servers. Bedrock's P4 code is compiled by proprietary toolchains provided by the switch and SmartNIC vendors.

A.4 Installation

We list the main steps to install Bedrock here. More details can be found in our GitHub repository.

- Install the BPF Compiler Collection (BCC) on end hosts for eBPF module compilation and loading.
- Install RNIC drivers to enable RDMA on end hosts.
- Install and setup the programmable switch and SmartNICs following the vendor instructions.

A.5 Experiment workflow

We briefly summarize the workflow of running experiments on Intel/Barefoot Tofino switches in Bedrock; detailed instructions can be found in the provided README in our GitHub repository. Note that all experiments in Bedrock share the similar workflow as described in the following.

1. **Compile P4 program:** Compile the P4 programs using Intel/Barefoot Tofino switch SDE.
2. **Run Bedrock or baseline:** Run P4 programs on the switch. Both Bedrock's programs (i.e., `switch/bedrock_*.p4`) and the baseline program (i.e., `switch/baseline.p4`) are provided.
3. **Load eBPF modules (for authentication experiments only):** Load Bedrock's eBPF module on the RDMA servers and clients for authentication experiments.
4. **Setup the logging server (for logging experiments only):** Setup and start the logging server for logging experiments.
5. **Compile and run RDMA servers, clients, and attackers:** All needed end host programs (i.e., RDMA server, RDMA client, and attacker) can be compiled with `make` in the corresponding folders. Folder and file names are summarized in Table 4.

A.6 Evaluation and expected results

We evaluate Bedrock in different attacking scenarios. The following describes the expected results:

Authentication: We uses attack S1 to demonstrate effectiveness of proposed source authentication in Bedrock. When the experiment starts, the server terminal will keep dumping the memory content. The attack can be launched by setting the victim client's QPN, PSN, and rkey in the attacker program. Without Bedrock (`baseline.p4`), the memory content will keep changing, indicating that the attacker illegally accesses the memory. By deploying Bedrock, the attack will be blocked and the memory content will remain the same.

ACL: Bedrock enables more flexible ACLs inside the network. In this experiment, when Bedrock is not started, all RDMA

Experiment	Folder	Server	Client	Attacker
Auth.	`authentication`	`server_auth`	`client_auth`	`client_attacker`
ACL	`authorization/attack_demo`	`server_acl`	`client_acl`	N/A
Mon.-BW	`monitoring/bw_exhaustion`	`victim`	`client`	`client`
Mon.-QP	`monitoring/qp_exhaustion`	`victim`	N/A	`attacker`
Log.	`logging/pythia_attack_demo`	`server`	`client`	`client`

Table 4: The folder and file names of end host programs for each experiment.

requests will get responses (printed on the client terminal). Bedrock enables operators to deploy new ACL rules and block RDMA requests violating the rules. If a request is blocked by Bedrock, the client will show `Completion status 12`.

Monitoring—bandwidth exhaustion: The effectiveness of the attack and defense is evaluated by the bandwidth usage of traffic from each client (refer to our Github repository for details). Results should be similar to Figure 6(b) in the main paper where the attack starts at t=2.7s and Bedrock mitigates the attack at t=5.4s.

Monitoring—QP exhaustion: In this experiment, attackers try to consume as many QPs (queue pairs) as possible to cause QP exhaustion. Without Bedrock, the attacker can keep creating queue pairs on the server until the server cannot allocate queue pairs anymore. With Bedrock, a single client can only consume a predefined number of queue pairs. When the attacker tries to ask for more, the IP address will be banned and no further RDMA traffic will be allowed from that user.

Logging: We demonstrate the logging system in Bedrock by detecting and mitigating the Pythia side-channel attack (see more details in the main paper). Running the experiment will output the accuracy of the attack both at the terminal and in the output folder `logging/pythia_attack_demo/output`. Without Bedrock, the accuracy can be as high as 95%, but Bedrock will detect and mitigate the attack.