USENIX Security '24 Artifact Appendix: GoFetch: Breaking Constant-Time Cryptographic Implementations Using Data Memory-Dependent Prefetchers

Boru Chen
UIUC

Yingchen Wang
UT Austin

Pradyumna Shome
Georgia Tech

Christopher W. Fletcher
UC Berkeley

David Kohlbrenner
University of Washington

Riccardo Paccagnella
Carnegie Mellon University

Daniel Genkin
Georgia Tech

A Artifact Appendix

A.1 Abstract

This artifact includes source code of reverse engineering experiments and key extraction attack PoCs in the research paper. The hardware requirement is an Apple M1 machine (more details in A.2.3). The software requirement is macOS 13 or 14 (more details in A.2.4). Reproducing reverse engineering experiments will take approximately 1 day. For attack PoCs, the time consumption is summarized in Table 1 of the research paper.

A.2 Description & Requirements

This artifact contains (i) reverse-engineering experiments that disclose the activation criteria of Apple DMPs and (ii) one pedagogical PoC (constant-time swap) and four real cryptography PoCs (Go's RSA Encryption, OpenSSL Diffie-Hellman Key Exchange, CRYSTALS-Kyber and CRYSTALS-Dilithium) that demonstrate how to leverage Apple DMPs to conduct chosen-input attacks against constant-time cryptographic implementations.

A.2.1 Security, privacy, and ethical concerns

Experiments are running on evaluator-owned machine under full control. No other machines are involved.

A.2.2 How to access

All source code is posted to our GitHub repository `https://github.com/FPSG-UIUC/GoFetch`. The stable reference is `https://github.com/FPSG-UIUC/GoFetch/releases/tag/usenix2024ae`.

A.2.3 Hardware dependencies

The hardware requirement is an Apple M1 machine. Other M-series CPUs (i.e. M1 Max, M2, M3, etc) should show similar reverse-engineering results with parameter tweak (e.g. cache size), while attack PoCs require engineering efforts to port to other Apple silicons.

A.2.4 Software dependencies

A default configuration of macOS (reverse-engineering experiments have support for Asahi Linux). Programs require `Rust` and `Clang` for compilation.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Run `xcode-select -install` to install Xcode command line tools. Run `curl -proto '=https' -tlsv1.2 -sSf https://sh.rustup.rs | sh` to install Rust.

A.3.2 Basic Test

Please follow **Set up Environment** and **DIT Bit Test (Quick Test)** in the README to check the functionality of reverse-engineering tools. We expect to see *Detect DMP signals!!* on M1/M2 no matter DIT bit is set or not and on M3 processors, setting DIT bit should result in *NO DMP signals!!*.

Please follow **Constant-Time Conditional Swap** in the README to check the functionality of attack PoC tool.

A.4 Evaluation workflow

Please checkout the README in the artifact for instructions of how to conduct and what to expect for each experiment. The experimental results could vary depending on different processors in Apple M-series and operating systems, but the figure trend/claim in the paper should maintain.

A.4.1 Major Claims

(C1): *We reverse engineer the DMP found on Apple CPUs and discover new activation criteria that prior work overlook. This is proven by experiments in Section 4 of the paper.*

(C2): *We develop new type of chosen-input attack with DMP assistance and demonstrate it with end-to-end PoCs. This is proven by experiments in Section 5-7 of the paper.*

A.4.2 Experiments

Please checkout the README in the artifact for experiment descriptions.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.