ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

NVLeak: Off-Chip Side-Channel Attacks via Non-Volatile Memory Systems

Zixuan Wang[⋆] Mohammadkazem Taram[♯⋆] Daniel Moghimi[†⋆]
Steven Swanson[⋆] Dean Tullsen[⋆] Jishen Zhao[⋆]

[⋆]UC San Diego [♯]Purdue University [†]UT Austin

A Artifact Appendix

A.1 Abstract

This artifact describes NVLeak, a collection of microarchitecture-level reverse-engineering tools and covert/side channel attack proof-of-concept, which exploit the microarchitecture design of Intel Optane DIMM. NVLeak also comes with a set of scripts to set up system environments, run experiments, collect results, and generate plots.

A.2 Description & Requirements

NVLeak artifacts are available online as a GitHub repo which contains reverse engineering, covert/side channel code, data parsing scripts, and documentation to use these tools. To reproduce the major claims in the main paper, we recommend using a server machine with Intel Optane DIMM, similar to *Server A* or *Server B* described in the main paper.

A.2.1 Security, privacy, and ethical concerns

This artifact does not exploit any security breaches on evaluators' machines. The reverse-engineering and covert/side channel code is run on the server machine with Optane DIMMs, and it is not destructive to evaluators' environments. The only code running on evaluators' machines is data parsing and plot generation scripts, which can run in the Docker images from NVLeak.

A.2.2 How to access

NVLeak code is hosted on GitHub[1].

A.2.3 Hardware dependencies

NVLeak exploits the microarchitecture designs of Intel Optane DIMMs and thus requires these DIMMs to reproduce the results presented in the main paper. An example server machine environment with Optane DIMM is shown in Table 1.

Table 1: NVRAM-equipped server system configuration.

Hardware	Configuration
CPU	Intel Xeon Gold 6230 20 Cores per socket, 2 sockets HyperThreading off
L1 Cache	32 KiB 8-way I-Cache, 32 KiB 8-way D-Cache, private
L2 Cache	1 MiB, 16-way, private
L3 Cache	27.5 MiB, 11-way, shared
DRAM	6 channels per socket DDR4, 16 GiB, 2666MHz
NVRAM	Intel Optane DIMM, 6 channels per socket 128 GiB, 2666 MHz Firmware: 01.02.00.5355

A.2.4 Software dependencies

NVLeak has a kernel module that compiles with Linux 5.4 or older versions (tested with 5.1 and 4.15). A newer kernel may have breaking changes to the filesystem APIs used by NVLeak and thus may fail the compilation. NVLeak requires `ndctl` (v67+) and `ipmctl` (v02.00.00.3885) to configure the Intel Optane DIMMs, which can be compiled and installed from their source code on GitHub. Additional NVLeak requires `e2fsprogs` (v1.46.4 or newer) to configure the Ext4 filesystem, and `sqlite3` (v3.31.1), `PMDK` library, `wolfSSL` (v4.2.0) for side channel attacks. NVLeak GitHub repo has more detailed documentation on installing and using these tools.

A.2.5 Benchmarks

NVLeak requires *NPPES NPI* dataset for SQLite side channels and provides a script to download this dataset.

A.3 Set-up

NVLeak provides a set of scripts to set up the server machine for experiments. Due to the space limit, we provide minimal instructions in this artifact appendix and describe the complete setup process in the NVLeak GitHub repo.

[1] https://github.com/TheNetAdmin/NVLeak/tree/588567e6ec30f2df9f260e60385031c94e94c75e

```
$ sudo -i su
# cd NVLeak/nvleak
# bash scripts/machine/machine.sh setup
# reboot
# bash scripts/machine/optane.sh reset
# bash scripts/machine/optane.sh setup \
      appdirect ni
# bash scripts/machine/optane.sh ndctl
```

Figure 1: Set up the Linux boot arguments and Optane DIMM operation modes.

```
$ ndctl list -u
[
    {
        "dev":"namespace1.0",
        "mode":"fsdax",
        "map":"dev",
        "size":"124.03 GiB (133.18 GB)",
        "uuid":"***",
        "sector_size":512,
        "align":2097152,
        "blockdev":"pmem1"
    },
    {
        "dev":"namespace0.0",
        "mode":"fsdax",
        "map":"mem",
        "size":"32.00 GiB (34.36 GB)",
        "sector_size":512,
        "blockdev":"pmem0"
    }
]
```

Figure 2: Optane DIMMs are successfully configured into the non-interleaved mode (the pmem1 device is around 128 GiB, a single DIMM's size), and the kernel boot argument memmap successfully creates an emulated PMEM device pmem0 using DRAM.

A.3.1 Installation

NVLeak provides scripts to set up the system, including Linux boot commands and Optane DIMM operation modes, as listed in Figure 1.

Each NVLeak experiment requires a different set of tools and Optane DIMM configurations. In general, each setup involves three steps: (1) Install required tools, e.g., sqlite3; (2) Configure Optane DIMMs and mount them as Linux devices; (3) Compile the source code in NVLeak. Please refer to NVLeak GitHub's documentation for more details.

A.3.2 Basic Test

To check if the setup takes effect, run ndctl and check if a non-interleaved PMEM device and an emulated PMEM device are created, as shown in Figure 2.

A.4 Evaluation workflow

Table 2: Major claims and corresponding results.

Figure	Type	Claims
2	Reverse Engineering	L1/L2 NVCache sizes, their block sizes, and WPQ size
4		L1/L2 NVCache set structures
5		Wear-leveling policy
6		Wear-leveling's trigger condition
7		Robustness of wear-leveling data migration
17		Detailed pointer chasing results on Server A
18		Reverse engineering results on Server B
9b-c	Covert Channel	Cross virtual machine covert channel performance and signal
10		Filesystem inode-based covert channel
12	Side Channel	Access patterns of SQLite executing different SQL code
13		Access patterns of SQLite executing ranged queries
14		Access patterns of PMDK key-value store
15		Detected function calls from wolfSSL library
16	Mitigation	Effectiveness and performance of the PMDK-based mitigation

A.4.1 Major Claims

As shown in Table 2, we have made the following four major claims in our main paper:

(C1): NVLeak is able to reverse engineer the Optane DIMM's microarchitecture designs, including WPQ size, NVCache set structures, and wear-leveling mechanisms.

(C2): NVLeak can establish covert channels based on the recovered off-chip microarchitecture to break virtualization and file system isolation.

(C3): NVLeak can establish side channels to leak sensitive information from applications that use NVRAM as storage or memory.

(C4): NVLeak can mitigate the recovered vulnerability by patching the PMDK library's memory allocator.

A.4.2 Experiments

The major NVLeak experiments can be categorized into four types, as listed below. NVLeak GitHub repo provides complete documentation to set up hardware/software environments and reproduce results. The GitHub repo also contains scripts to collect and parse data, generate plots, and compile a LaTeX PDF with all plots organized as in the main paper.

(E1): Reverse Engineering *[1 human-hour + 6 compute-hours + 18 GiB disk]:*

Steps: Configure the Optane DIMM into non-interleaved mode, then compile and insert the NVLeak kernel module, and finally run the scripts to execute all experiments. NVLeak also provides Slack integration to send experiment progress to the Slack channel configured by the user.

Results: Reproduce Figure 2-7 and Figure 17-18.

Docs: See *docs/reproduce/ReverseEngineering.md* in the NVLeak GitHub repo for more details.

(E2): Covert Channel *[1 human-hour + 16 compute-hours + 32 GiB disk]:*

Steps: Configure the Optane DIMM into non-interleaved mode, and create two separate Linux PMEM devices for the sender and receiver. Then compile the user space proof-of-concept, including QEMU and KVM-unit-tests. And finally, execute NVLeak's scripts to run experiments.

Results: Reproduce Figure 9-10.

Docs: See *docs/reproduce/CovertChannel.md* in the NVLeak GitHub repo for more details.

(E3): Side Channel *[2 human-hours + 1 compute-hours + 1 GiB disk]:*

Steps: Create two separate Linux PMEM devices for the attacker and victim. Then download the NPPES dataset and initialize the SQLite database. And finally, run NVLeak scripts to start experiments.

Results: Reproduce Figure 12-15.

Docs: See *docs/reproduce/SideChannel.md* in the NVLeak GitHub repo for more details.

(E4): Mitigation *[2 human-hour + 1 compute-hours + 1 GiB disk]:*

Steps: Download and compile the PMDK library, then execute NVLeak scripts to evaluate the effectiveness and performance of the mitigations described in the main paper.

Results: Reproduce Figure 16.

Docs: See *docs/reproduce/SideChannel.md* in the NVLeak GitHub repo for more details.

A.5 Notes on Reusability

NVLeak is able to exploit Optane DIMM's microarchitecture, and the user can establish attacks based on these hardware designs. But NVLeak is not limited to Optane DIMM as NVLeak is not bound to any Optane-specific hardware or software. One example is that NVLeak can run on DRAM, as shown in our main paper, Figure 10b.

We envision that NVLeak can be used to exploit future memory devices' microarchitecture designs, such as new memory products based on the Compute Express Link (CXL) technology. In fact, we have repurposed NVLeak to reveal PCIe performance characteristics (not shown in this paper) by attaching an FPGA to PCIe and using MMIO to map the FPGA memory for NVLeak to access. We hope NVLeak can facilitate future memory security research for not just NVRAM but even broader memory technologies.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2023/`.