Midas: Systematic Kernel TOCTTOU Protection

Atri Bhattacharyya, *EPFL;* Uros Tesic, *Nvidia;* Mathias Payer, *EPFL*

https://www.usenix.org/conference/usenixsecurity22/presentation/bhattacharyya

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

For Midas, we present an artifact including the source code and binaries for the prototype based on Linux, an exploit which demonstrate that Midas mitigates a real CVE, and benchmarks for evaluating Midas' performance, and scripts which simplify the process. In the following sections, we describe the artifact, its requirements and how to run it, and what the expected results are. Visit the project website https://hexhive.epfl.ch/midas for more details.

A.1 Description

The primary artifact for this paper is the code implementing Midas on the Linux kernel (v5.11), available on GitHub. We also provide a disk image suitable for recreating experiments from this paper, containing the kernel as both source code and as compiled binaries. The disk image contains the CVE exploit used to test correctness in the paper, all benchmarks evaluated in the paper, and scripts to run these. This image allows recreation of all emperical evidence presented in the paper's evaluation. Finally, we provide further information on the project website including a detailed description of the artifact, its contents, how to run it and expected outputs.

- Source code: https://github.com/HexHive/midas
- Disk image: https://zenodo.org/record/5753026
- Project website: https://hexhive.epfl.ch/midas

A.1.1 Hardware Dependencies

You can run the disk image within a QEMU virtual machine to test functionality. The host machine requies around 100GiB free disk space and at least 8GiB memory. You should run the disk image on a real machine for performance tests. Our Midas prototype supports machines with 64-bit x86 processors, and the results in the paper were obtained on a machine with an Intel i7-9700 CPU. Further, the real machine requires an empty 1TiB disk, and a EUFI-enabled motherboard. In both setups, a SSD is preferred for storage, as it leads to faster compilation should you choose to re-compile the kernel. Evaluating the Nginx benchmark requires a second, networked machine to act as a load generator.

A.1.2 Software Dependencies

Running the Midas disk image requires a guest operating system which supports running QEMU. The image was tested on QEMU version `4.2.1` on a machine running Ubuntu 20.04 with Linux kernel version `5.4.0-88-generic`. Other virtualization software should also be supported, but the instructions focus on QEMU. Running the disk image on real hardware requires no special software support, apart from a tool to write the image to a disk. On Linux, we can use `dd`.

A.2 Installation

The installation procedure includes downloading and uncompressing the provided compressed disk image, then either running a VM directly from this image, or by writing the image to a disk and booting from it.

On Linux, the following command extracts the image.

```
pv ae.img.xz | unxz -T <num threads> > ae.img
```

The uncompressed disk image can then either be run with QEMU, or written to a real disk. To run with QEMU, an example command is shown below.

```
qemu-system-x86_64                    \
  -m 4G                               \
  -cpu host                           \
  -machine type=q35,accel=kvm         \
  -smp 4                              \
  -drive format=raw,file=ae.img       \
  -display default                    \
  -vga virtio                         \
  -show-cursor                        \
  -bios /usr/share/ovmf/OVMF.fd       \
  -net user,hostfwd=tcp::2222-:22     \
  -net nic
```

To run on real hardware, copy the image to a real disk using the command shown below, then install into the machine and start it.

```
dd if=ae.img of=/dev/<disk> bs=100M
```

A.3 Experiment Workflow

The experimental workflow compares the modified Midas kernel with the baseline Linux kernel. Detailed steps are available on the website at https://hexhive.epfl.ch/midas/docs/ae.html. You can validate the artifact by executing the following steps:

- Check that the code modifications described in the paper correspond to the code.
- Compile the code to re-create the kernel binary.
- Run a script to check that a CVE exploit is mitigated, as claimed in the paper.
- Run scripts to execute the benchmarks presented in the paper, to verify their reported performance.

For the CVE exploitation test, the dmesg output must be checked to ensure that Midas prevents exploitation. For the performance experiments, the results must be compiled and compared to get the Midas' relative performance. The general workflow is:

- boot with the correct kernel (baseline or Midas),
- run the script for the benchmark/CVE exploit,
- reboot with the other kernel, and
- run the same script again.

A.4 Expected Results

Midas is evaluated to demonstrate effective mitigation of double-fetch bugs with low overhead. The artifact enables you to verify this claim, that the prototype provides the claimed protection and that it performs as claimed. We demonstrate the first property by including checks in the kernel and running an exploit for CVE-2016-6516 to demonstrate its mitigation. The remaining benchmarks measure performance, either as operations per second or as time taken to finish each operation. Below, we describe how to interpret the outputs of running the exploit and benchmarks.

Midas protects the kernel against double-fetch bugs, and in particular mitigates an exploit for CVE-2016-6516. In our prototype, you will execute the exploit with and without Midas' protections. When run with the baseline kernel, the exploit is triggered, and the string `"Triggered bug: CVE-2016-6516!"` will be printed to `dmesg` output. With the Midas kernel, the string is never printed.

We also run kernel-intensive benchmarks which demonstrate that Midas has a low runtime overhead. Our artifact also contains the performance benchmarks used for testing Midas' performance. The benchmarks must be run separately with both the baseline and Midas kernel. We include a script to plot the relative performance vs. the baseline kernel. Midas' performance is strongly dependent on the CPU used for evaluation, and exact performance values can vary significantly. However, we expect the trends of performance across benchmarks to roughly follow the following limits.

- Microbenchmarks see results in line with paper.
- NPB benchmarks experience 0-5% overhead, and should follow the numbers from the paper.
- PTS benchmarks - openssl, git, pybench, redis see an overhead <1%.
- PTS benchmarks - apache sees a overhead < 10-15%.
- PTS benchmarks - IPC benchmark sees overhead < 5%.
- Nginx shows a constant overhead as request size changes, until the network link is saturated.

The setup for breaking down Midas' overhead is complicated, and omitted from this artifact.

A.5 Artifact meta-information

- **Program:** NASA Parallel Benchmarks (NPB), Phoronix Test Suite (PTS), Nginx, the Linux kernel, and exploits for CVE-2016-6516. All benchmarks and code are publicly available, and are installed in the provided disk image.
- **Binaries:** The disk image provides the compiled Linux kernel (v5.11) with and without Midas' protections.
- **Hardware:** For functionality evaluation, one machine with 100GiB free disk space, and QEMU (version 4.2). For results reproduction, one machine with modern Intel x86 CPU, and a free 1TiB disk. In both setups, a SSD is preferred.
- **Run-time state:** The disk image includes a program for fixing CPU frequency, eliminating run-time variance. This only works on native hardware, not QEMU.
- **Metrics:** NPB workloads report execution rate. PTS workloads report either execution time or operation rate. Nginx reports both request rate and throughput.
- **Output:** Most benchmarks and tests output to a console.
- **Experiments:** Experiments have been prepared within the disk image, and can be run using provided scripts.
- **How much time is needed to prepare workflow (approximately)?:** 3-4 hours, on a machine with an SSD.
- **How much time is needed to complete experiments (approximately)?:** For performance evaluation, approx. 8 hours.
- **Publicly available?:** All code is publicly available.
- **Code license:** GPL v2.0
- **Archived (provide DOI or stable reference)?:** DOI 10.5281/zenodo.5753026 available at https://zenodo.org/record/5753026.