
# USENIX Security '25 Artifact Appendix: Shadows in Cipher Spaces: Exploiting Tweak Repetition in Hardware Memory Encryption

Wei Peng[*] Yinshuai Li[*] Yinqian Zhang[†]
Research Institute of Trustworthy Autonomous Systems
Department of Computer Science and Engineering
Southern University of Science and Technology

## A Artifact Appendix

### A.1 Abstract

Our paper introduces the CIPHERSHADOW attack, which exploits the reuse of the same tweak value in the 64-byte memory blocks of the Hygon CSV v1, v2, v3 memory encryption mechanisms. This reuse leads to successful decryption through 16-byte ciphertext replacement, and the characteristic that identical 16-byte ciphertexts have the same plaintext. CIPHERSHADOW can tamper with the program's control flow and data flow by substituting encrypted memory after interrupting the VM at an attacker-chosen point in the program execution. This artifact includes two attacking scenarios: a malicious SSH login implemented to CSV v1 and v2, and a one-shot ciphertext leakage of the MNIST dataset implemented to CSV v1, v2, and v3.

### A.2 Description & Requirements

This artifact comprises four main experiments: 1) Gadget scanning; 2) Proof of Concept (PoC) of SSH-malicious login; 3) Page location; 4. PoC of one-shot ciphertext leakage. The experiments must be conducted on Hygon processors. For the SSH-malicious login and page location, the processor must be capable of enabling VMs with CSV v1 or v2. For the one-shot ciphertext leakage, the processor needs to be able to enable VMs with CSV v1, v2, or v3.

#### A.2.1 Security, privacy, and ethical concerns

Observing the ciphertext does not pose a security issue, but replacing 16-byte aligned ciphertext blocks may cause the VM to crash if the content of the program replacement is not correctly chosen or the timing of the replacement is incorrect.

#### A.2.2 How to Access

This artifact are publicly available on Zenodo: https://doi.org/10.5281/zenodo.15614377 and GitHub: https://github.com/pw0rld/CSV-CipherShadow.git.

#### A.2.3 Hardware dependencies

This attack must be executed on a Hygon-series CPU with at least CSV v1 support. Our experiments were conducted on the following processors:

- **Hygon C86 5285 CPU with CSV v1 support.**
- **Hygon C86 7390 CPU with CSV v2 support.**

#### A.2.4 Software Dependencies

Our experiments rely on the following software that supports enabling CSV VMs:

- **QEMU** (`csv-devel-6.2.0-v1`)[*]
- **Linux Kernel** (`OLK-6.6`)[†]
- **OVMF** (`csv-stable202202-v1`)[‡]

### A.3 Set-up

After installing and building the compatible versions of QEMU, the kernel, and OVMF, you can use the following commands to start CSV.

```
git clone github.com:pw0rld/CSV-CipherShadow
cd CSV-CipherShadow
./make_vm_img.sh
sudo ./start-qemu.sh
```

[*]These authors contributed equally to this work.
[†]Corresponding Author: yinqianz@acm.org

[*]https://gitee.com/anolis/hygon-qemu/tree/csv-devel-6.2.0-v1/
[†]https://gitee.com/openeuler/kernel/tree/OLK-6.6/
[‡]https://gitee.com/anolis/hygon-edk2/tree/csv-stable202202-v1/

### A.3.1 Basic Test

We provide a simple memory observation experiment to demonstrate the existence of this vulnerability. If the vulnerability exists, you should observe repeated 16-byte aligned ciphertext patterns within 64-byte blocks for identical plaintexts.

```
cd Observation_memory && make
insmod ob.ko
dmesg
```

## A.4 Evaluation workflow

The experiments we have prepared include application fingerprinting, SSH malicious login, and a one-shot ciphertext leakage attack.

### A.4.1 Major Claims

**(C1):** *We implemented a gadget scanner (Section 5) to find gadgets in target programs that can manipulate program control flow and data flow, by replacing aligned ciphertexts of 16 bytes within 64 bytes.*

**(C2):** *We demonstrate that a program fingerprint can be constructed based on the pattern of whether the 16-byte ciphertexts within a 64-byte block are identical, which helps in locating program pages (Section 6.1).*

**(C3):** *We demonstrate that* CIPHERSHADOW *hijacks program control flow and tampers with function return values by replacing 16-byte ciphertexts, enabling malicious login to SSH services (Section 6.2).*

**(C4):** *We demonstrate that in the CSV memory encryption mechanism, the characteristic of 16-byte aligned identical ciphertexts within 64 bytes having the same plaintext can lead to one-shot ciphertext leakage attacks (Section 6.3).*

### A.4.2 Experiments

**(E1): Gadget Scanning**:

**How to:** Select the sensitive functions in the application you want to attack and search for gadgets to help tamper with the control flow or data flow, thereby achieving the purpose of tampering with the return value.

**Preparation:** Download the necessary Python dependencies.

```
pip install -r requirements.txt
```

**Execution:** Execute the gadget scanner script, the first parameter is the target program's binary file, the second parameter specifies the function for finding gadgets, and the third parameter configures specific values of registers and memory at the function entry.

```
python gadget_search.py <Target Binary>
    <Target Functions> <Initial State
    Value>
```

**Results:** You can see the results of the gadget scan in the terminal, including the number of illegal and potential control flow and data flow gadgets, along with the corresponding assembly content of each gadget.

**(E2): SSH-Malicious Login [10 minutes]**:

**How to:** The attacker is able to log into the TEE VM using an incorrect password through the SSH service. In this experiment, we demonstrate that the encryption mechanism of CSV allows for replacing 16 bytes within a 64-byte block and successfully decrypting, and by utilizing the ciphertext pattern to locate pages, we hijack the return value of `sys_auth_passwd` function by substituting the code for failed login attempts with that of successful ones.

**Preparation:** First, the attacker needs to preprocess the SSH binary to identify the characteristic features of the page containing the *sys_auth_passwd* function. Then, using page faults, the attacker obtains suspicious VM pages. After filtering a specific set of VM pages, the attacker extracts their memory features and compares them with the preprocessed features. A match indicates that the page is the target page. If multiple matches occur, further methods are required for precise identification, such as leveraging the register page features discussed later.

**Notice:** If the machine freezes when starting CSV, you can run the reload.sh script in the repository. This script will reload the kernel and remove previous cached configurations.

**Execution:** The tool automatically targets the currently running CSV VM, repeatedly attempts SSH logins until the SSH login interface is identified, and modifies the login logic to validate successful authentication.

```
sudo python3 analysis_csv.py
```

**Results:** The attacker is able to log into the server using an incorrect password.

**(E3): Page location [10 minutes]**:

**How to:** In this experiment, we use a benchmark consisting of a set of assembly instructions with special patterns. We extract their characteristics using single-stepping. This allows us to observe the execution of instructions and locate the target page.

**Preparation:** First, the attacker needs to compute the register characteristics of a specific assembly code segment within the target page. This process requires manual analysis using GDB. In this experiment, we use a 14-bit value to record the change patterns of different registers. If a register changes compared to the previous state, the corresponding bit is set to 1. In this experiment,

the register feature we use is: [0x1, 0x80, 0x40, 0x10, 0x8, 0x4, 0x4, 0x8, 0x10, 0x40, 0x80, 0x1].

**Execution:** This command triggers a timer interrupt and scans a designated Guest Physical Address (GPA) to extract its physical page characteristics. The provided parameters include the physical address and the timer interval.

```
./poc single_step_page 0x10af9e000 10
dmesg -c > dmesg.log
```

**Noticed:** Please note that due to differences in machine APICs, testers may need to identify a suitable APIC Timer value on their own machines. On our machines, the effective range is between 7 and 10.

**Results:** The attacker can observe the register behavior of the page through its GPA, and leverage this feature to locate the page by matching it with the preprocessed binary.

**(E4): One-Shot Ciphertext Leakage [5 minutes]**:

**How to:** For CSV v3, read and write protection has been implemented on encrypted memory, preventing attackers from performing ciphertext replacement attacks. However, we can infer and recover the training set for the MLP-Classifier model by utilizing the characteristic that identical 16-byte aligned plaintexts have identical ciphertexts within a 64-byte block through ciphertext patterns.

**Preparation:** Since we do not have physical access to the CSV v3 machine, we simulated the one-shot memory collection process in CSV v2. After the training set was loaded into memory, we completed the dump of the encrypted memory. The file `leak.bin` contains the collected training set memory data.

**Execution:** Run the leakage attack script, which includes three steps: recovering images, training the model, and validating the success rate of recovering images.

```
python leakage-attack.py
```

**Results:** The restored images will be in the `of_dir` folder, and the terminal will output the success rate of the image recovery.

## 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/usenixsec2025/.