

Implementation of Meltdown Attack Simulation for Cybersecurity Awareness Material

Eka Chattra¹, Obrina Candra Briliyant²

^{1,2}Politeknik Siber dan Sandi Negara, Bogor, Indonesia

Article Information

Received:
Accepted:
Published:
DOI: 10.33555/ejaict.v...

Corresponding Author:

Eka Chattra and
Obrina Candra Briliyant
Email:
eka.chattra@poltekssn.ac.id
obrina@poltekssn.ac.id

ISSN 2355-1771

ABSTRACT

One of the rising risk in cybersecurity is an attack on cyber physical system. Today's computer systems has evolve through the development of processor technology, namely by the use of optimization techniques such as out-of-order execution. Using this technique, processors can improve computing system performance without sacrificing manufacture processes. However, the use of these optimization techniques has vulnerabilities, especially on Intel processors. The vulnerability is in the form of data exfiltration in the cache memory that can be exploit by an attack. Meltdown is an exploit attack that takes advantage of such vulnerabilities in modern Intel processors. This vulnerability can be used to extract data that is processed on that specific computer device using said processors, such as passwords, messages, or other credentials. In this paper, we use qualitative research which aims to describe a simulation approach with experience meltdown attack in a safe environment with applied a known meltdown attack scheme and source code to simulate the attack on an Intel Core i7 platform running Linux OS. Then we modified the source code to prove the concept that the Meltdown attack can extract data on devices using Intel processors without consent from the authorized user.

Keywords: Cybersecurity, Cache Memory, Meltdown Attack, Processor, Out-of-Order execution

1. Introduction

Cyber security awareness is very important to maintain confidentiality, integrity, and ensure service availability in the context of organization's digital services. Security is guarded in the form of personal data such as documents, messages, texts and even passwords used to access a service. One of the most effective way to provoke cyber security awareness is to experience the risks, the vulnerabilities and the exploits first-hand. In 2017, vulnerabilities were found in computer processors, especially Intel processors, which could be exploited using an attack called Meltdown[1]. Any Intel processor released since 1995, except for Intel Itanium and Intel Atom before 2013, could potentially be affected by this attack[2]. Meltdown was able to allow an attack to read memory addresses in the kernel that should not have been accessible to the user[3][4]. Meltdown can even be used on devices with admin access rights to get data from the virtual machine (VM) guest account[5]. The meltdown attack is capable of exploiting the out-of-order execution that occurs in modern processors[1]. Out-of-order execution is an optimization feature that maximizes the utilization of all units of the central processing unit (CPU) core, so it can run instructions in parallel and can help improve processor performance[6].

Considering the impact of this exploitation, Kohli [7]conducted research in 2018 by simulating a Meltdown attack using an environment that was run on a virtual machine, with

reference to the stages of the Meltdown attack carried out by *Lipp et al.* This is intended as educational material, learning media and courseware as cybersecurity awareness material for students or researchers to know and experience the vulnerabilities exploited by the Meltdown attack. Courseware is a tools that can be used for learning purposes and act as a physical means of conveying subject matter[8]. In this context, this study implements a simulation stage of the Meltdown attack based on the scheme proposed by *Lipp et al* with reference to the Meltdown simulation conducted by *Kohli*. The simulation goals is to displays the output of secret data containing 8 characters placed in the processor's cache, as a prove of the concept of Meltdown attack.

1.1 Meltdown Attack

Meltdown is a cyberattack by means of exploiting a processor vulnerability that allows an attacker to bypass the isolation boundary between the application and the operating system (OS) so that data or information retrieval can occur when an application is running. The name was derived from the fact that the attack was able to "melt" the insulation protection which the processor hardware should be able to provide. Meltdown affects almost all Intel's processors produced since 1995, except for Intel Atom before 2013. Meltdown can exploit and infect all types of computer devices, including smartphone devices and cloud services [1]. Logically, a Meltdown attack occurs when it is about to read data from memory. Meltdown designed to exploit the out-of-order execution that today's modern generation processors applied. During the out-of-order execution, the processes carried out in memory are stored through registers and cache. If the out-of-order is not executed, the instruction is discarded from memory and registers, but the content of the memory are not lost in the cache. Meltdown takes advantage of this condition by using a side channel to check the availability of locations in the memory in the cache [9]. The Meltdown attack consists of 3 main stages as follows: (1) the content of the memory location chosen by the attacker are loaded into the register so that they can be accessed by the attacker, (2) a transient instruction on the CPU access the cache-line, based on the confidentiality of the content in the register, (3) the attack uses the weakness of the side channel on the processor (flush and reload) to determine the cache paths that are accessed and the data stored in the memory register[2].

1.2 Out-of-Order Execution

Each CPU core has several execution units, each of which is capable of carrying out different types of operations. In managing to keep the unit stable while working, the CPU can see various instructions that will be carried out afterwards and start executing on each CPU core while waiting for the next instruction. The out-of-order execution is a modern technology for a processor to be able to work optimally from the execution units available in the CPU [10]. In an out-of-order execution, instructions are fetched in the order according to the compiler generated. Processors that have an out-of-order execution function do not have to wait for previous instructions to complete to execute the next instruction, it can perform parallel executions.

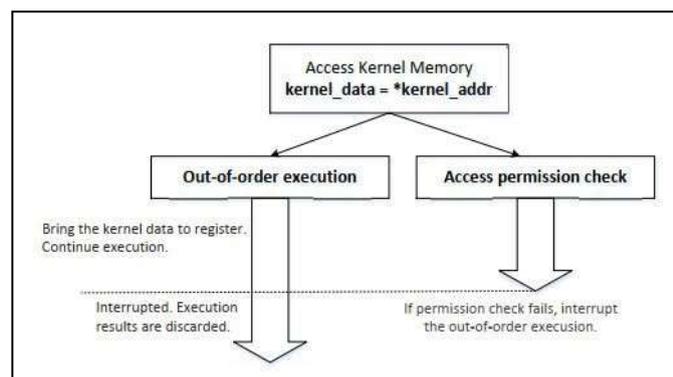


Figure 1. Out-of-Order Process on Modern Processors

As figure 1 depicted, in the out-of-order execution, the instructions will be executed in the order of data availability provided. As for doing so, after the processor takes program instructions from memory, then the instructions are sent to the instruction queue. When waiting for the queue, the instruction is allowed to leave the queue to carry out the next activity. Every time there is a call, the instruction will leave the queue to be executed. After that, instructions are sent to the unit accordingly. If all instructions command were to return to the registration result, then the result of the instruction are written to the register file.

1.3 Operating System

The operating system (OS) responds to a given process by managing the processes and system resources required by the user and the processes as mandated in the program. As the general platform of software, the OS performs basic tasks such as controlling and allocating, prioritizing process requests, controlling input and output devices, facilitating network systems and managing file systems. Most OS have applications that provide an interface to the resources managed by the OS. The OS functions as a hardware input/output management system as well as an interface in the use of software services. Linux is an OS that is free and open source. This makes Linux a very flexible and customizable OS. Most Linux distributions are designed for general use on desktop computers and network servers, but there are also distributions that are specific to different purposes and environments.

1.4 Simulation as a Learning Media

The simulation approach requires knowledge of the operation of the desired processes and output in a controlled environment. This information must be captured by the simulation model to represent the behaviour of the planned processes. But the level of detail included in the simulation will depend on the goal of the simulation [11]. For the purpose of cyber security awareness through meltdown attack simulation, the detailed unrelated processor works are not needed. So does the detailed information about how the program e.g. the source code are introduced to the victim's system. By reducing irrelevant or dynamic variables, the simulation can be constructed at a higher level to raised awareness caused by cyberattacks especially computer hardware exploits. After the framework of the simulation has been created, various computer configurations then may be created and much experiments can be perform according to several attack scenarios. And because the simulation's variable are controlled and safe in the virtual machine, they can be repeated and modified as needed so that all ground truth information can be identified and analysed.

2. Methods

The research method used in this research is qualitative research methods. This method is used to identify and understand the meaning behind the visible data used or proposed in previous research. The attack simulation design is based on the attack scheme conducted by *Lipp et al.* However, the stages in simulating a Meltdown attack are adjusted according to the research needs, which are carried out into several steps. These steps can be seen in Figure 2. These steps will also be the main reference for the design of the courseware.

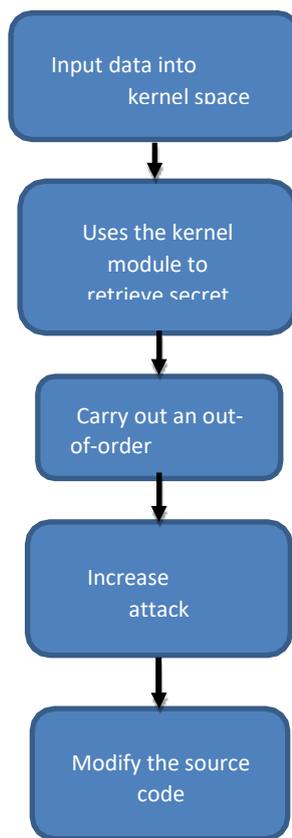


Figure 2. Meltdown Attack Simulation Scheme Planning

After understanding the visible data in the form of scheme and or concept of the research, then we used experiment method to gain a feasible approach of how to paint a picture of the danger of meltdown attack. The experimentation involved modifying source code so that the simulation can tests attack on user spaces. The modification also added some iteration so that the address reading can be conducted up to 1000 time. This loop procedure can improve the attack capabilities as such so that the attack's output can extract all available characters in the cache address. Quick comparison of research conducted with the references used is presented in Table 1.

Table 1. Research Comparison regarding Meltdown Attack

Comparison Variable	Previous Research	This Research
Virtual OS	Ubuntu 16.04	Ubuntu 18.04
Virtual Machine	Vbox 5.2	Vbox 6.0
Testing	Do not test attack on user space Does not test read address loops	Test attack on user space Test the loop reading address from 0-1000
Output	One character	All available characters

3. Result and Discussion

This chapter contains an explanation of what was done in the designed scheme along with the results obtained during the implementation process based on the experiments carried out.

3.1 Preparing the Attack Simulation

The first step for simulating an attack is carried out based on the need for the implementation stage. The attack simulation process requires a virtual laboratory environment. In this research, the environment will be run using Oracle Virtualbox 6.0, which uses the Ubuntu 16.04 operating system with an Intel core i7 processor. In order to simulate a Meltdown attack, the thing to consider is to ensure that the data was stored in kernel space (virtually). The data stored in this research is a string consists of 8 characters and stored as hardcode. These characters can later be seen or printed through a series of attack simulations.

```
static char secret[8] = {'P','A','S','S','W','O','R','D'};
static struct proc_dir_entry
*secret_entry; static char*
secret_buffer;
```

In some conditions, to increase the attack power, it is optional to use assembler code in the source code. Because Meltdown attack has a significant effect on a large number of repetitions that are carried out to find out the value entered into the cache. In this study, we tried 1000 times of iterative process. To simulate the output of the attack, modification of the original source code[7] were conducted to be able to print the secret characters that are retrieved from the cache.

```
// Prints all characters at the secret
value unsigned long kernel_data_addr
=0xfed0000; int k = 0;
for (k = 0; k < 8; k++) {
    Meltdown_attack(kernel_data_addr)
    ;
    kernel_data_addr += 1;
}
```

Note that the unsigned line long `kernel_data_addr = 0xfed0000` statement refer to the secret data address in the kernel module.

3.2 Implementation of Attack Simulation

After preparing the simulation environment, the next stage is implementing the attack simulation steps.

3.2.1 Input Data into Kernel Space

In the attack simulation concept, confidential data is stored in the kernel space indicating that the stored data can be read by a vulnerability caused by Meltdown. To be able to store confidential data, the kernel module was used as a library in the kernel. After compiling, the source code only needs to be run with the command `"./(filename)"` just like running common source code on Linux OS. There are several conditions for running a Meltdown attack simulation carried out on the kernel module[7]:

- a. To simulate an attack, the address of the target confidential data is required. The target address will be obtained when inserting the kernel module into kernel memory. To perform these steps, run the command `make` to compile the kernel module. When finished compiling, you will see the addition of files in the form of kernel modules. It can be seen in Figure 3 before compiling the kernel source code, and Figure 4 after compiling the kernel source code

```
Makefile
MeltdownAttack.c
MeltdownExperiment.c
MeltdownKernel.c
```

Figure 3. Before compiled

```
Makefile
MeltdownAttack
MeltdownAttack.c
MeltdownExperiment
MeltdownExperiment.c
MeltdownKernel.c
MeltdownKernel.ko
MeltdownKernel.mod.c
MeltdownKernel.mod.o
MeltdownKernel.o
modules.order
Module.symvers
```

Figure 4. After compiled

- b. After compilation, insert the kernel module into kernel memory with the command `sudo insmod MeltdownKernel.ko` so that the address of the secret data can be stored in the kernel. When the process is successful, it can continue to find the address of the secret data stored in the kernel message buffer using `dmesg` command.

```
[07/10/20]seed@chattrra:~/../Simulasi-Meltdown1$ sudo insmod MeltdownKernel.ko
[07/10/20]seed@chattrra:~/../Simulasi-Meltdown1$ dmesg | grep 'secret data address'
[ 1596.077407] secret data address: f9d51000
[54948.287789] secret data address: f9ed0000
[81660.411171] secret data address: f9c9a000
```

Figure 5. Process of obtaining Secret Address

Seen in Figure 5, the `dmesg | grep 'secret data address'` command can also be done so that search results can be found quickly and you can immediately see the address of secret data used in the kernel environment. In the figure there are 3 addresses of confidential data because the process carried out in this study is repeated. The address used is the most recent secret data address with a value of `f9c9a000`.

- c. Stored confidential data must be in the processor's cache so that the attack simulation can run smoothly.

3.2.2 Utilized the Out-of-Order Execution

Once a confidential data address has been obtained, the address is entered in the source code

```
if (sigsetjmp(jbuf, 1) ==
0) {
Meltdown(0xf9c9a000);}
```

The source code is executed by utilizing an out-of-order execution where the stored data is entered into cache memory via a source code snippet as shown below.

```
int fd = open("/proc/secret_data",
O_RDONLY); if (fd < 0) {
perror("open"
); return -
1;}
int ret = pread(fd, NULL, 0,
0); if (ret < 0) {
perror("pread"
); break;}
```

3.2.3 Optimizing Attack

To be able to print secret characters stored in the kernel, it is necessary to increase the power of the attack by using a looping reading process of the kernel memory so that the value of the stored characters can be found and printed out. In this research, testing was carried out with the number of iterations of the same array as many as 256 at different loops. The test is described in Table 2 to see the results of the increase in attack power.

Table 2. Testing on Repetition Reading Secret Addresses

The number of repetitions of the read address	The number of loop 8 bits arrays	Result
0	256	Printed value 0
10	256	All values are printed
100	256	All values are printed
500	256	All values are printed
1000	256	All values are printed

3.2.4 Attack Simulation with Source Code Modifications

The Meltdown attack simulation can be run at this stage, but we decide to experiments and try to modified the source code so that the printed results can proved that the Meltdown attack simulation can retrieve all the data (all the characters) in the kernel memory. Note that the unsigned long line `kernel_data_addr = 0xf9c9a000`; address corresponds to the address on each device after inserting the kernel module into the kernel.

```
[07/10/20]seed@chattr:~/.../Simulasi-Meltdown1$ ./MeltdownAttack
The secret value is 80 P
The number of hits is 981
The secret value is 65 A
The number of hits is 857
The secret value is 83 S
The number of hits is 984
The secret value is 83 S
The number of hits is 990
The secret value is 87 W
The number of hits is 334
The secret value is 79 0
The number of hits is 972
The secret value is 82 R
The number of hits is 941
The secret value is 68 D
The number of hits is 982
```

Figure 6. Modification of Meltdown Attack Simulation

Based on Figure 6, it can be seen that the Meltdown attack simulation can be carried out according to *Kohli (2018)* to print out a character of the secret data in the kernel. Also, by modifying the source code we can prove that the Meltdown attack can retrieve then print out all available characters of the secret data in the kernel.

4. Conclusion

Based on the attack simulation stages carried out, it can be concluded that a courseware for simulating a Meltdown attack consists of 4 steps that can be done by the learners. Those steps are: (1) input secret data into kernel space, (2) utilized the out-of-order execution, (3) optimizing attack, and (4) attack simulation with/without source code modification. However, previous to conducting such steps, the simulation environment must be setup and communicated between the instructor and the learners so that the context of the attack can be

understood. The attack simulation process requires a virtual laboratory environment with the correct hardware installed (e.g. Intel processors). Also, the virtual laboratory environment must ensure that the secret data was stored in kernel space. Simulated Meltdown attack provide by the steps can proves that the Meltdown attack can exploit the out-of-order execution built-in on the Intel Core i7 processor by issuing secret data values stored in the kernel memory. Further research can be explored regarding the notion of experimenting with the possibilities of other modification in the source code to found unknown capabilities of the attack, and thus helps identified other vulnerabilities in the target system. In terms of raising cybersecurity awareness, we encourage researchers to study the best way to explain about cyberattacks, cyber risks and threats in an effective way to learners and common audiences.

5. References

- [1] M. Lipp *et al.*, “Meltdown : Reading Kernel Memory from User Space,” 2017.
- [2] M. K. Follow, “Meltdown and Spectre, explained,” 2018.
- [3] A. Prout *et al.*, “Measuring the Impact of Spectre and Meltdown,” *2018 IEEE High Perform. Extrem. Comput. Conf. HPEC 2018*, pp. 1–5, 2018, doi: 10.1109/HPEC.2018.8547554.
- [4] P. Company, “White Paper How the Meltdown and Spectre bugs work and what you can do to prevent a performance plummet THE I / O PROFILING Meltdown and Spectre : The facts,” 2018.
- [5] J. Sianipar, M. Sukmana, and C. Meinel, “Moving sensitive data against live memory dumping, spectre and meltdown attacks,” *26th Int. Conf. Syst. Eng. ICSEng 2018 - Proc.*, pp. 1–8, 2019, doi: 10.1109/ICSENG.2018.8638178.
- [6] Y. Kao and J. Huang, “High-Performance NAND Flash Controller Exploiting Parallel Out-of-Order Command Execution,” pp. 160–163, 2010.
- [7] K. Kohli, “Meltdown Attack Lab,” no. February, pp. 1–15, 2018.
- [8] Rusman, *Belajar dan Pembelajaran Berbasis Komputer Mengembangkan Profesionalisme Guru Abad 21*. Bandung: Alfabeta, 2012.
- [9] H. Grover and D. Agrawal, “Design and architecture of Intel ’s core i7 processor,” vol. 2, no. X, 2014.
- [10] B. A. Ahmad, “Real time Detection of Spectre and Meltdown Attacks Using Machine Learning,” no. April, 2020, [Online]. Available: <http://arxiv.org/abs/2006.01442>.
- [11] M. E. Kuhl, J. Kistner, K. Costantini, and M. Sudit, “Cyber attack modeling and simulation for network security analysis,” *Proc. - Winter Simul. Conf.*, pp. 1180–1188, 2007, doi: 10.1109/WSC.2007.4419720.