CSCI 5403 / CYBR 5300
Fall 2022

Team B Progress Report

Katrina Siegfried, Xen Hasnat, Matthew Taylor, Aaliyah Siburg

1. Setting up virtual machines

To begin, we will first be downloading and setting up Oracle's VM VirtualBox. Then, we will install the individual three virtual machines that we will be using in order to execute the penetration tests and vulnerability scans in the next steps. First, we download and install the Kali Linux Oracle Virtual Appliance (OVA) and set it up with VirtualBox as two separate VM's, one will be the detection machine and the other will be the attacker machine. We will then download and install the Ubuntu 20.04 LTS distribution to VirtualBox and ensure it is configured with the 5.8.0 Linux kernel. The three VM's will all be configured to use a Host-Only networking configuration with promiscuous mode disabled for all machines except for the detection Kali VM which will be configured to allow all.

We are then able to continue to the next step, which is setting up a testing lab that includes executing the Dirty Pipe vulnerability.

Detailed instructions can be found here:
https://github.com/siegfrkn/CSCI5403_CVE20220847_Detection/tree/main/Instructions

2. Setting up penetration testing lab with vulnerable target distributions for Dirty Pipe

Once the Virtual Machine is set up, we will need to create a pipe - a pipe provides a unidirectional interprocess communication channel. Data from the write end of a pipe can be read from the read end of a pipe. Example using an anonymous pipe: `echo hello | wc -c` where the output of the first process is passed into the pipe and then wc -c process uses it as an input.

To exploit the vulnerability, we need to:

- Create a pipe.
- Fill the pipe with arbitrary data to set the `PIPE_BUF_FLAG_CAN_MERGE` flag in all ring entries.
- Drain the pipe leaving the flag set in all `struct pipe_buffer` instances on the `struct pipe_inode_info ring`.
- Splice data from the target file opened with `O_RDONLY` into the pipe from just before the target offset.
- Write arbitrary data into the pipe, this data will overwrite the cached file page instead of creating a new anonymous `struct pipe_buffer` because `PIPE_BUF_FLAG_CAN_MERGE` is set.

There are a few limitations of this exploit:

- The threat actor must have read permissions because it is necessary to splice() a page into a pipe; but write permissions are NOT necessary.
- The offset must not be on a page boundary because at least one byte of that page must have been spliced into the pipe.
- The write cannot cross a page boundary because a new anonymous buffer would be created for the rest.
- The file cannot be resized as the pipe has its own page fill management and does not tell the page cache how much data has been appended.

3. Performing vulnerability exploitation for the Dirty Pipe exploit

We will be launching the attacks using this exploit from the Kali VM's to the vulnerable Ubuntu target.

A small executable program will be written that opens the pipe and splices the new information into the cache buffer as explained above. This will allow us to reliably and easily reproduce the exploitation.

The requirements to successfully launch our attack on the target machine are:

- Machines must be on the same network or the attacking machine must be able to gain network access to the target machine.
- The attacking machine requires readonly access
-

We expect that any Linux Kernel versions with 5.8 will be vulnerable to the exploit

4. Determine if the dirty pipe exploit can be detected by defensive threat software in real time

Two machines, at minimum, are required to set up the detection exercise – the Kali VM which will be used as the attacker machine, and the Ubuntu VM which will be used as the victim machine. These two virtual machines have been set up and networked by the team. Additionally, a third VM can be set up for monitoring and generating logs but has not yet been set up. This exercise will have five general steps as the team has outlined and intends to perform this week: 1) information gathering, 2) scanning of infrastructure setup, 3) attacker exploitation and target defense / scanning, 4) documentation and information gathering regarding the attack and defense / scanning, and 5) reporting of the real-time vulnerability detection.

The scanning infrastructure is of particular interest in this project as the objective is to detect exploitation of the dirty pipe in real time. As this vulnerability only affects the cache, it is cleared on power cycle and therefore is only detected via patterns in logging, or potentially in real time. Scanning tools that may warrant some investigation include

OpenVAS or Datadog to help watch for splice system calls, Wireshark to help detect suspicious incoming network traffic, and custom scripts to monitor user permission changes.