International Journal of Internet, Broadcasting and Communication Vol.14 No.3 17-21 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.3.17

IJIBC 22-3-3

A study on Dirty Pipe Linux vulnerability

[1]Saurav Tanwar, [2]Hee Wan Kim

[1]Student, Division of Computer Science & Engineering, Sahmyook Univ., Korea
[2]Prof., Division of Computer Science & Engineering, Sahmyook Univ., Korea
tsaurav18@gmail.com, hwkim@syu.ac.kr

Abstract

In this study, we wanted to examine the new vulnerability 'Dirty Pipe' that is founded in Linux kernel. how *it's exploited and what is the limitation, where it's existed, and overcome techniques and analysis of the Linux kernel package. The study of the method used the hmark[1] program to check the vulnerabilities. Hmark is a whitebox testing tool that helps to analyze the vulnerability based on static whitebox testing and automated verification. For this purpose of our study, we analyzed Linux kernel code that is downloaded from an open-source website. Then by analyzing the hmark tool results, we identified in which file of the kernel it exists, cvss level, statistically depicted vulnerabilities on graph which is easy to understand. Furthermore, we will talk about some software we can use to analyze a vulnerability and how hmark software works. In the case of the Dirty Pipe vulnerability in Linux allows non-privileged users to execute malicious code capable of a host of destructive actions including installing backdoors into the system, injecting code into scripts, altering binaries used by elevated programs, and creating unauthorized user profiles. This bug is being tracked as* ***CVE-2022-0847*** *and has been termed "****Dirty Pipe"[2]*** *since it bears a close resemblance to* ***Dirty Cow[3]****, and easily exploitable Linux vulnerability from 2016 which granted a bad actor an identical level of privileges and powers.*

Keywords*, Dirty pipe, Dirty cow, Linux vulnerability, CVE-2022-0847, Hmark*

1. Introduction

As reported by Max Kellermann, the person who identified the Dirty Pipe vulnerability. this vulnerability is similar to an older vulnerability disclosed in 2016, Dirty Cow (CVE-2016-5195), which has been actively exploited by malicious actors since then. And according to the experts, this vulnerability is easier to exploit than Dirty Cow was. Dirty Pipe, as the name suggests, makes use of the pipeline mechanism of Linux with malicious intent. Pipeline is an inter-process communication mechanism that takes the output of the first process and passes that to the second process as its input[4]. Piping is an age-old mechanism in Linux that allows one process to inject data into another. The vulnerability is due to an uninitialized "pipe_buffer.flags"

Manuscript Received: May. 11, 2022 / Revised: May. 14, 2022 / Accepted: May. 17, 2022
Corresponding Author: hwkim@syu.ac.kr
Tel:*** - **** - ****
Professor, Division of Computer Science & Engineering, Sahmyook Univ., Korea

variable, which overwrites any file contents in the page cache even if the file is not permitted to be written, immutable, or on a read-only mount, including CD-ROM mounts. That is because the page cache is always ritable by the kernel, and writing to a pipe never checks any permissions.

It allows local users to gain root privileges on any system with publicly available and easily developed exploits. It is a unidirectional and inter-process communication method in which one process takes input from the previous one and produces output for the next in the line. Dirty Pipe takes advantage of this mechanism combined with the splice function to overwrite sensitive read-only files for instance, /etc/passwd, which can be manipulated to gain a no-password root shell. Table 1 is explained a brief introduction to CVE-2022-0847[4].

Table 1. Overview of CVS-2022-0847

Associated CVE ID	CVE-2022-0847
Description	A vulnerability in the Linux kernel that allows overwriting data in arbitrary read-only files.
Associated ZDI ID	–
CVSS Score	7.8 High
Vector	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Impact Score	–
Exploitability Score	–
Attack Vector (AV)	Local
Attack Complexity (AC)	Low
Privilege Required (PR)	Low
User Interaction (UI)	None
Scope	Unchanged
Confidentiality (C)	High
Integrity (I)	High
availability (a)	High

2. Exploit and Limitation

Before talking about exploitability, let's talk about the constraints to be considered regarding this vulnerability. In order to exploit it:

- The threat actor must have read permissions as, without it, they would not be able to use the splice function.
- The offset must not be on the page boundary, because at least one byte of that page must have been spliced into the pipe
- The write process cannot cross a page boundary because a new anonymous buffer would be created for the rest.
- The file cannot be resized, because the pipe has its own page fill management and does not tell the page cache how much data has been appended.

Here are the required steps to exploit:

- First, initialize a pipe.
- Then we populate the pipe with arbitrary content and then clear it. This step is necessary to set the PIPE_BUF_FLAG_CAN_MERGE flag.
- Once done, we splice() data from the target file into the pipe just before the offset.
- Finally, we overwrite the cached file page with write().

3. Measurement Tools and Analysis Method

For verification, we used the Vulnerable Code Clone Detection (VCC Detection)[5] is an approach for the scalable detection of vulnerable code clones, which is capable of detecting security vulnerabilities in large software programs efficiently and accurately.

The Whitebox-testing hmark tool was used to check whether the vulnerability existed or not. hmark is a preprocessor for the target program, and the hmark generates a .hidx file with every function of the specified program. After getting .hide file analysis was performed.

The Dirty pipe vulnerability is found only in linux-kernel, it effects endpoints running Linux with a kernel version 5.8 or higher.[5] This includes a multitude of devices running android 12 and Linux. Therefore, the analysis was conducted only with Linux package. VCC Detection provides you with various types of analysis. The number of detected vulnerable functions are shown in Figure 1, followed by the top 3 vulnerable files and the top 3 CVE occurrences. The origin of vulnerabilities indicates the original repository in which the detected vulnerable functions had been reported and patched[6].

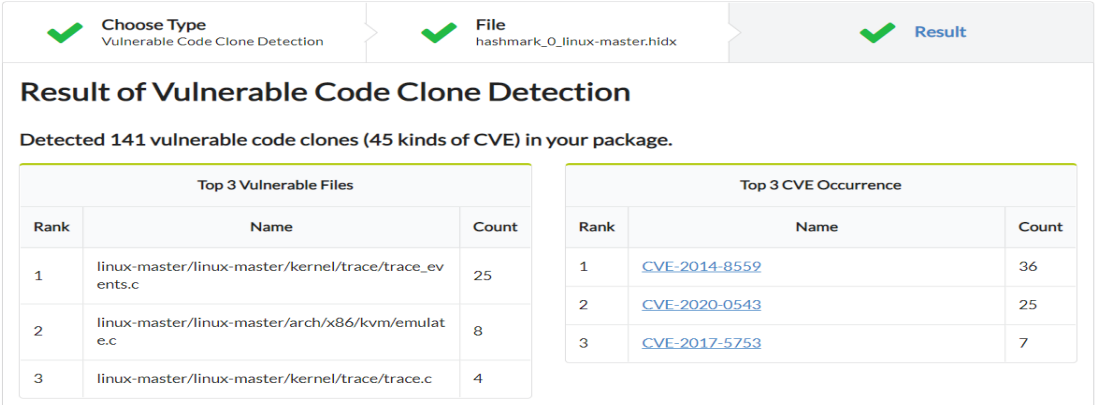

Figure 1. Browsing the Result of Linux master package

We can also find the yearly distribution of CVEs, the distribution of CVSS, and the distribution of CWEs in Figure 2, 3. It was easy to find out vulnerability by yearly distribution. we just checked each year and the highest number of vulnerabilities counted in the year 2017. In addition, we can easily check by just direct clicking on the year 2022 to examine dirty pipe vulnerability or any other specific year.

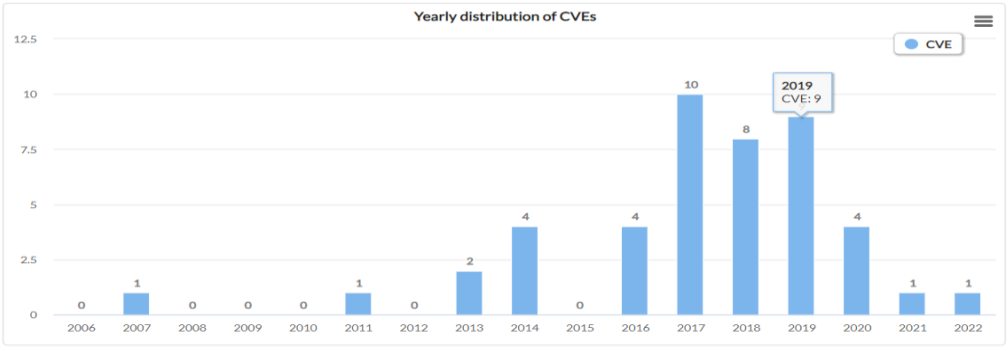

Figure 2. Yearly distribution of CVEs

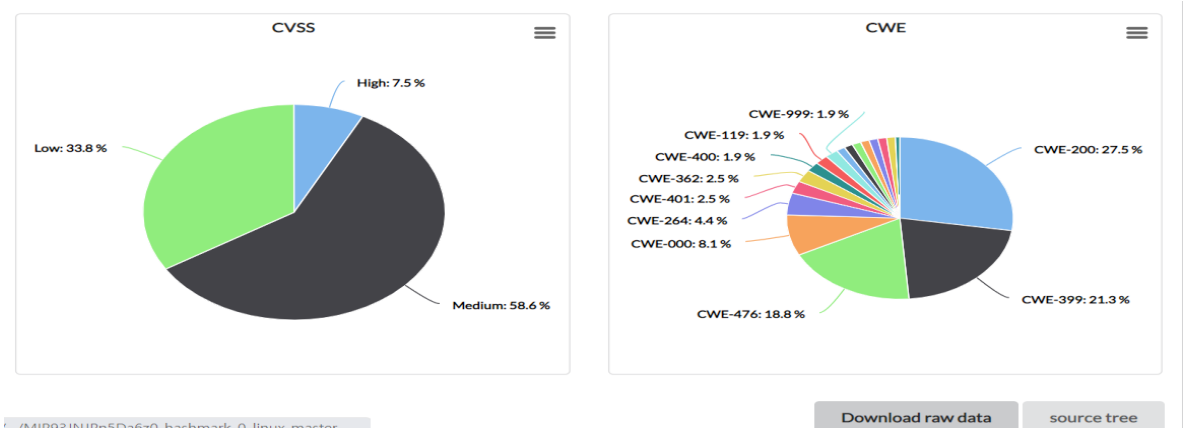

Figure 3. Yearly distribution of CVSS and CWE

Below is the source tree in which paths and files which contain vulnerable function are illustrated in Figure 4. Each leaf node corresponds to a file, and the numbers in the leaf nodes denote the number of detected vulnerable functions in the file[8].

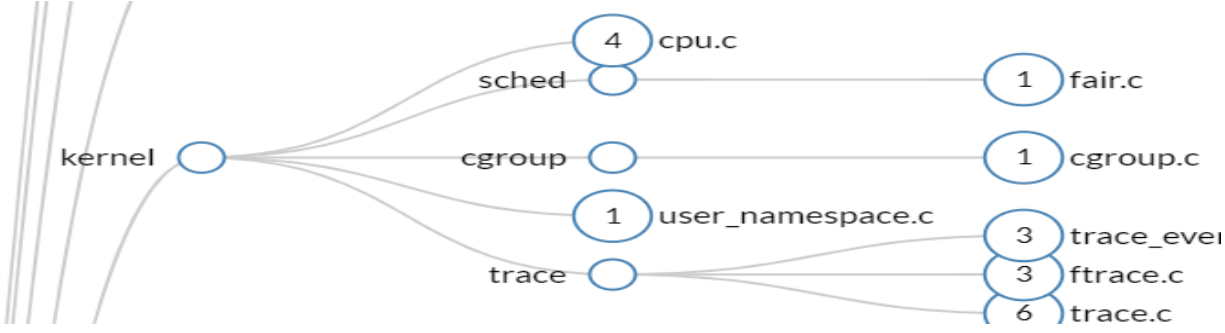

Figure 4. Source Tree

Clicking on a leaf node displays detailed information of the vulnerability and its patch in Figure 5.

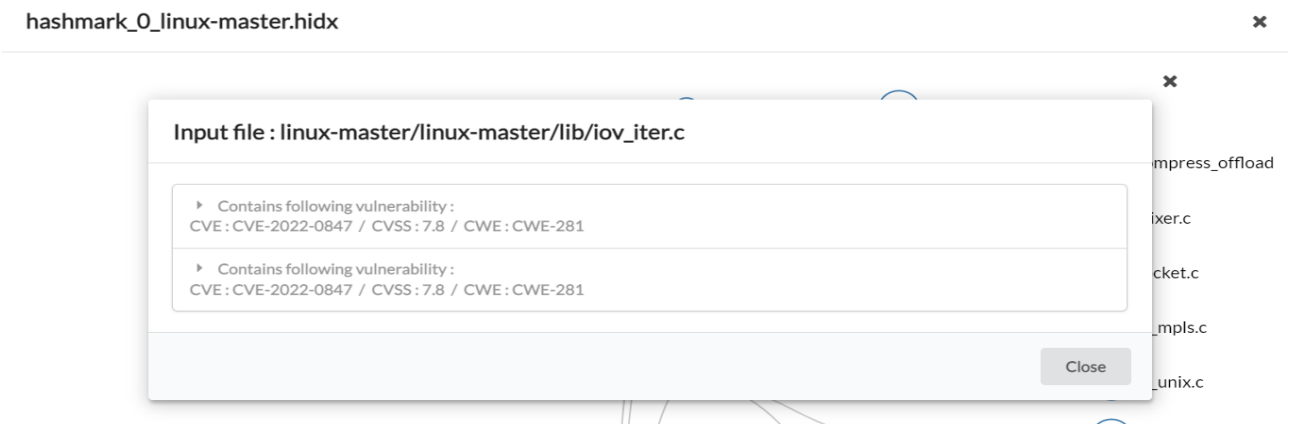

Figure 5. Detailed Information of vulnerability

4. Countermeasures and Conclusion

Currently, many embedded devices are being released and many industries are developing based IoT. It will be of great help in preventing vulnerabilities in advance through hmark in IoTcube's whitebox-testing. Through this test, the hmark tool is largely based on the vulnerability of the code itself and vulnerability detection through CVE. The output of hmark Is remarkable and easy to understand with various types of statistics combinations. I think tools like this become important to analyze packages. These days software has many vulnerabilities therefore hmark could be useful to detect it. on the other hand, I think machine learning or deep learning helps to explore vulnerabilities predicting by applying algorithms to detect patterns of vulnerabilities and software to automatically respond to detected vulnerabilities. Many studies have been already conducted to patch it.

As we know to analyze the source code and binary of the software one technique is static analysis, dynamic analysis, and hybrid according to the characteristics can be classified by lead analysis. In addition, software penetration tests could be used in the order to find vulnerabilities, because software penetration testing makes software more secure to find vulnerabilities through a legitimate hacking process. In the case of general vulnerability diagnosis, potential. It's a process to find issues related to security, and it's a penetration test is to check how the vulnerability you've explored can be used. a process is added. To this end, search, scanning, attack, and access. It goes through a process of maintenance. These penetration tests measure software security. Apply the penetration test at the item level and risk the result. Many security threats can be managed through analysis. I think it could help predicting these worst points with above techniques and artificial intelligence techniques to remove vulnerabilities in the field of security.

REFERENCES

[1] Internet of Things simplified, https://iotcube.net/
[2] References to Advisories, Solutions, and Tools, https://nvd.nist.gov/vuln/detail/CVE-2022-0847
[3] Dirty Cow, https://www.cs.toronto.edu/~arnold/427/18s/427_18S/indepth/dirty-cow/index.html
[4] How To fix the Dirty Pipe Vulnerability in Linux Kernel – CVE-2022-0847, https://thesecmaster.com/how-to-fix-the-dirty-pipe-vulnerability-in-linux-kernel-cve-2022-0847/
[5] How To Mitigate CVE-2022-0847 (The Dirty Pipe Vulnerability), https://www.ivanti.com/blog/how-to-mitigate-cve-2022-0847-the-dirty-pipe-vulnerability
[6] Secure Software Supply Chain Management, https://iotcube.net/userguide/manual/hmark