Study of the Dirty Copy on Write, a Linux Kernel Memory Allocation Vulnerability

Tanjila Farah, Rummana Rahman, M. Shazzad Hosain
Department of Electrical & Computer Engineering
North South University
Dhaka, Bangladesh
e-mail: {tanjila.farah, rummana.rahman, shazzad.hosain}@northsouth.edu

Delwar Alam, Moniruz Zaman
Department of Software Engineering
Daffodil International University
Dhaka, Bangladesh
e-mail: {delwaralam, m.shojol80}@gmail.com

***Abstract*—Dirty Copy on Write also known as Dirty COW is a Linux based server vulnerability. This vulnerability allows attackers to escalate the file system protection of Linux Kernel, get root privilege and thus compromise the whole system. Linux kernel version 2.6.22 and above are affected by this vulnerability. The patch for this vulnerability has been released very recently. Thus most of the Linux servers are still vulnerable to this attack. This study focuses on the techniques of Dirty COW vulnerability and its impact of the servers of newly emerged IT country such as Bangladesh.**

Keywords-linux kernel; dirty COW; memory allocation; vulnerability; user privilege; root exploit

I. INTRODUCTION

All Now a days the popularity of web applications is the higher than ever because the large number of companies conducting business online. Web applications have become one of the most prevalent platforms for information and services due to their accessibility everywhere. This platform is now used for critical services such online transaction, news publication, personal information storage and sharing and many more. Because of the increasingly pivotal role of this platform, it has become a popular and valuable target for security attacks. Web applications around the world face an average of 27 attacks per hour. If the attack is automated (i.e. using tool) a target web application may experience up to 25,000 attacks per hour [7]. According to white hat web application attacks represent the greatest threat to an organization's security with 40% of breaches in 2015 [3]. According to their analysis 55% of the Retail Trade sites, 50% of Health Care/Social Assistance sites, and 35% of Finance/Insurance sites are always vulnerable. Whereas only 16% of the Retail Trade sites, 18% of Health Care/ Social Assistance sites, and 25% of Finance/Insurance sites are rarely vulnerable. Conversely, Educational Services is the best performing industry with the highest percentage of rarely vulnerable sites (40%) [3]. Arts, Entertainment, and Recreation are the next best industry with 39% of sites in rarely vulnerable category [3]. No web Application is immune to attacks as there is a new attack technique evolving every day. And due to this unprecedented increase in the scale and intensity of sophisticated web application attacks it is essential to study the different aspects of attacks carefully.

Due to the critical nature of the services provided through web platform by various enterprises, the web applications have complex infrastructure. This infrastructure includes a client side and a server side. The web server is powerful computer with specialized operations. Web servers run on operating system (OS). The most popular OS's for web server are Linux and Windows. Around 47.8% of servers in the world uses Linux OS and 32.8% servers use windows server [4]. A server may host one or many web applications. Web servers are expensive, thus through small scale web hosting, many company uses the same server to host web applications. Large companies might use their own server to host their web applications. These servers are constant target of attackers. In this day and age of targeted attacks and zero-day exploits, servers need the same amount of security as the client's side, if not more. The attacks on web servers may occur due to outright coding errors or from Operating System error [5]. Most of today's vulnerabilities with web servers, though, are attributed to the creativity of the hacker community to find ways for the web servers to handle data in ways the web server developer did not intend. One such vulnerability of Linux operating system is "Dirty Copy On Write (Dirty COW)" [1]. This vulnerability is found in various versions and distributions of Linux. This vulnerability existed in Linux OS's for a long time but has been recognized recently. This paper aims on detailed survey of the affect of Dirty COW on the web servers of Bangladesh vulnerability. This paper also intends to conduct a detailed analysis of the method of Dirty COW attack.

This paper is organized as following sections: Section II presents an overview of web system. In Section III overview of Linux and Linux kernel in presented. An analysis of the Dirty COW attack is described in Section IV. An analysis of the current affect of the Dirty COW attack is presented in Section V. The paper concludes in VI.

II. OVERVIEW OF WEB SYSTEM

A web system consists of web browser, firewall, and host server. The host server includes web, application, and database server and firewall as shown in Figure 1. On the server side, a web application receives user inputs via HTTP requests from the client through browser and interacts with local file systems, back-end databases or other components for data access and information retrieval. It uses HTTP

978-1-5386-0403-8/17/$31.00 ©2017 IEEE

(Hypertext Transfer Protocol) to serve the files that form Web application to users, in response to their requests. The web server is includes static data (i.e. html, css) and apache, redix or other server software.

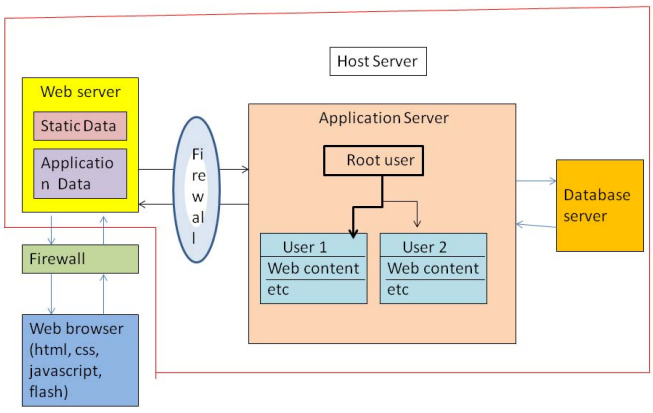

Figure 1. Generalized architecture of web application.

The host server includes both one root and multiple client users. The root user defines permissions for the client users so that one client cannot access files of clients. Each user is identified with a user id (uid), group id (gid), and groups.

The host servers can the categorized in three classes: Shared web hosting, Virtual Dedicated Server, and Dedicated hosting service. In Shared web hosting multiple web applications can be hosted. Virtual Dedicated Server also known as a Virtual Private Server (VPS), divides server resources into virtual servers, where resources can be allocated in a way that does not directly reflect the underlying hardware. VPS will often be allocated resources based on a one server to many VPSs relationship. The users may have root access to their own virtual space. In dedicated hosting service the user gets his or her own Web server and gains full control over it (user has root access for Linux/administrator access for Windows). Linux generally accepted to have an overwhelming majority over Windows servers. Companies such as Google use more than 15,000 Linux servers.

In web hosting services, the web applications belong to customers. For security to the host servers is an important concern. The level of security that a web hosting service offers is extremely important. Web hosting servers can be attacked by malicious users in different ways, including uploading malware or malicious code onto a hosted web application. These attacks may be done for different reasons, including stealing credit card data, launching a Distributed Denial of Service Attack (DDoS) or spamming. This study focuses on Dirty COW vulnerability which is Linux Kernel vulnerability. Linux operating system is discussed in later sections. There are other vulnerabilities in other OS's but that is out of the scope of this paper.

III. LINUX AND LINUX KERNEL

Linux is an open source operating system. This OS is cost effective and scalable compared to other server OS's. It is the fastest performing OS with the smallest resource footprint and most prevalent operating systems for web hosting. Linux operating system has number of discrete Linux distributions such as Red Hat Enterprise Linux (RHEL), CentOS, Fedora, Ubuntu, Debian, and OpenSuse. This distributions are similar except few changes in file system layouts, configuration settings, update mechanisms, and bundled configuration.

Each operating system uses a kernel. Windows, Mac OS X, and Linux all have kernels, and they are all different. The kernel grunts work of the operating system. The kernel talks to the hardware and software, and manages the system's resources. It talks to the hardware via the drivers that are included in the kernel. If an application running in the system has a requirement (i.e. change volume setting of the speakers), the request is processed by the kernel. The kernel is highly involved in resource management. Kernel deals with the allocation of memory space to run an application. It optimizes the usage of the processor to make the process faster. It also aims to avoid deadlocks, which are problems that completely halt the system when one application needs a resource that another application is using.

The Race condition technique, is a linux kernel vulnerability that miss uses the mechanism used by linux kernel to avoid dealdlock to implement the Dirty COW attack on a Linux based server.

A. Linux Memory Allocation

The OS memory allocation is divided as physical memory and virtual memory. Physical memory is present in the machine, and virtual memory is the address space. Virtual memory makes the system appear to have more memory than it actually has by sharing it between competing processes as they need it. Virtual memory manages memory management subsystem. This management subsystem ensures each process has its own virtual space and virtual address spaces are completely separate from each other so that a process running one application cannot affect another. It also provides mechanisms to protect certain areas of memory against writing. Virtual memory provides memory mapping which maps the contents of a file are linked directly into the virtual address space of a process. It also deals with shared virtual memory. While several processes in the system running the same root files (i.e. boot, proc, root, etc) rather than making several copies of the root files one in each processes virtual address space, it is better to have only one copy in physical memory and all of the processes running the file share. Kernel works with virtual addresses that are mapped to physical addresses by the memory management hardware. This mapping is defined by page tables, set up by the operating system. The relation between virtual addresses and physical addresses is given by page tables.

Whenever a command is executed, the file containing it is opened and its contents are mapped into the processes virtual memory. This is done by modifying the data structures describing this processes memory map and is known as memory mapping. If a process needs to bring a virtual page into physical memory and there are no free physical pages available, the operating system kernel must

make room for this page by discarding another page from physical memory.

Whenever a command is executed, the file contain When an process is loaded into memory the operating system needs to allocate pages. These will be freed when the process has finished executing and is unloaded. Another use for physical pages is to hold kernel specific data structures such as the page tables themselves. The mechanisms and data structures used for page allocation and de-allocation are perhaps the most critical in maintaining the efficiency of the virtual memory subsystem. The free\area vector is used by the page allocation code to find and free pages. The whole buffer management scheme is supported by this mechanism and so far as the code is concerned, the size of the page and physical paging mechanisms used by the processor are irrelevant.

For page allocation to a new process, page allocation code attempts to allocate a block of one or more physical pages. When a process is executed, the contents of the executable process must be brought into the processes virtual address space. The same is also true of any shared libraries that the executable process has been linked to use. The executable file is not actually brought into physical memory; instead it is merely linked into the processes virtual memory. Then, as the parts of the program are referenced by the running application, the image is brought into memory from the executable image.

The attack techniques discussed in this paper manipulates kernel memory allocation technique to implement the attack and escalate the privilege for the user.

B. Linux File Permission

Linux implements "Discretionary Access Controls" (DAC), which relies on a framework of users and groups to grant or deny access to various parts of the OS. The granted permission can be read-only, write-only, execute, write-execute, read-write, read-execute, read-write-execute and no permission. For example, a client user should be able to perform any action (read, write or execute without root users permission. Depending on the permission given by root user, a client user can perform specific or all action. Otherwise any malicious user with root permission may take over all the web application under one host server.

C. Race Condition

Race condition vulnerability is a flaw that produces an unexpected result when the timing of action impacts other actions. The steps of race condition are shown in Figure 2. In an OS environment operations work in multithread meaning one operation is completed and then the next operation starts and ended as shown in the correct behavior in Figure 2. The difference between these two operations is so small, that from users prospective these operations seem working in parallel.

In special cases, one operation might start before the previous operation is completed. As shown in Figure 2, task 1 starts and then soon after task 2 starts and finishes. Then task 1 finishes. So the output of the operation stays unexpected.

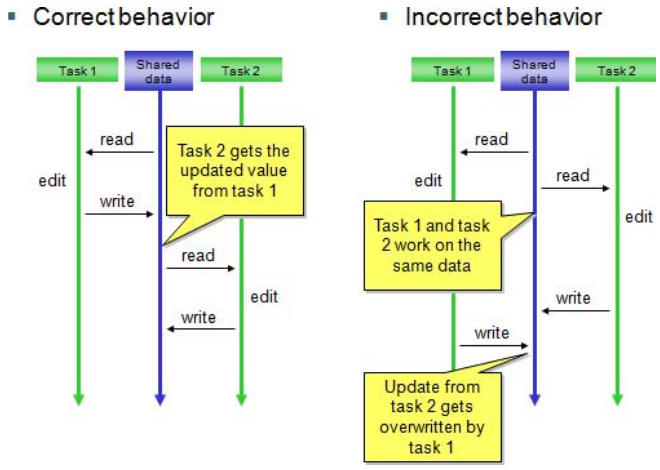

Figure 2. Race condition.

Using this hypothesis, race condition works. Through this attack, the attacker aims to gain root permission by overwriting the /etc/passwd/file [6]. This file holds all user login and permission. Same hypothesis is used in Dirty-COW attack.

IV. Analysis of Dirty COW Attack

The hypothesis of Dirty COW vulnerability targets the part of the Linux Kernel that is responsible for memory mapping. Linux uses the "Copy on Write" (COW) approaches to reduce unnecessary duplication of memory objects [6]. If you are a programmer, imagine you have the following code: let a = 'COW' and b = a. Even though there are two variables here, they both point at the same memory object -- since there is no need to take up twice the amount of RAM for two identical values. Next, the OS will wait until the value of the duplicate object is actually modified: b += 'Dirty'. At this point, Linux will allocate memory for the new, modified version of the object, read the original contents of the object being duplicated ('COW'), perform any required changes to it (append 'Dirty'), and write modified contents into the newly allocated area of memory. These steps tricks the memory mapper to write the modified contents into the original memory range instead of the newly allocated area, such that instead of modifying memory belonging to "b" we end up modifying "a".

A. Dirty COW Code Analysis

In a Linux system Root privileges means the complete access to all files and commands including ability to modify the system in any way desired and to grant and revoke access permissions (i.e., the ability to read, modify and execute specific files and directories) for other users, including any of those that are by default reserved for root [5]. A client user is not allowed to have root privilege. The Dirty COW vulnerability escalate the root privilege for the user. For example a Dirty COW attack on the /etc/passwd may change the user login id to root login id. The steps of this attack are as follow.

1) Checking initial user id: In a linux command line, writing the command "id" shown the current logged in user's

user id as uid=1001 (test), group id as gid= 1002 (test), and the groups=1002 (test) as shown in Figure 3. Here test is the user name of the example client user. user id and group id, in Linux system is different for every user. The user id starts from 1001 except only root users uid=0(root) [5].

```
test@ubuntu:/home/seed/labs/dirty_cow$ id
uid=1001(test) gid=1002(test) groups=1002(test)
test@ubuntu:/home/seed/labs/dirty_cow$
```

Figure 3. Checking user if before Dirty COW attack.

2) Running attack script: Next step is running the Dirty COW attack script. The client user saves the attack script in the server as per the client privilege the runs and the script from client side only\cite{etcpasswrd}. In the attack shown Figure 4, the Dirty COW script is save in the file attack\passwd file in the "test" users account. The user executes the script and the exits the account. Again logging in under the "test" user and writing the "id" command returns uid= 0(root), gid= 1002 (test), and the groups=0(root),1002(test). Thus the test user account is identified as root user account by the system. This gives the test user privilege to write on root files.

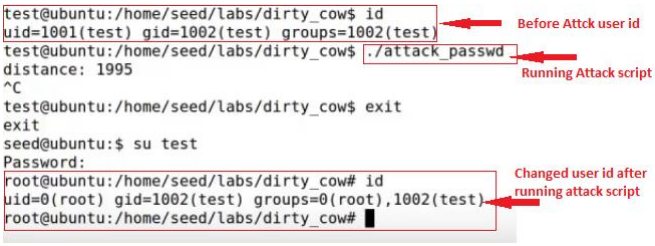

Figure 4. Checking user if before Dirty COW attack.

3) The source code of Dirty COW exploit: The exploit start with the main() function. In this function the root file is called as read-only (O\RDONLY flag) as shown in Figure 5. The next is executing the "mmap" command to create a new mapping in the virtual address space of the etc/passwd process. This means the opened file gets a new address in the memory. In Linux system when mapping a file to the memory address, the file is not placed in RAM but it is 'connected' to the place where the file is stored. Thus the file is read from the main memory. Using the "mmap" command the kernel is directed to choose a new memory address which has the size same as the size of the etc/passwd file. In that address a copy of etc/passwd is open tagged as MAP\PRIVATE. The MAP\PRIVATE parameter allows to edit the copied file. This makes the mapped memory field 'copy -on write' (COW of 'Dirty COW'). Every time the exppoit is ran, it will create a copy of the original file (i.e. /etc/passwd) to write to. The original file should not be edited. It also sets the pointer where the the memory should start reading.

```
int main(int argc, char *argv[])
{
  pthread_t pth1,pth2;
  struct stat st;

  // Open the file in read only mode.
  int f=open("/zzz", O_RDONLY);

  // Open with PROT_READ.
  fstat(f, &st);
  map=mmap(NULL, st.st_size, PROT_READ, MAP_PRIVATE, f, 0);
```

Figure 5. Figure 1: The code for Dirty COW exploit: calling the root function

In the next two lines the methods 'madviseThread' and "procselfmemThread" is called. These methods are used for tricking the kernel into writing the root\file as shown in Figure 6. The 'madviseThread' method calls the 'MADV\DONTNEED' parameter with the root\file. This parameter directs kernel that the address range assigned for the root file "/etc/passwrd" is not needed anymore and the kernel can free the resources associated with it. This address range is now considered 'Dirty' (of 'Dirty COW') or empty.

```
  // We have to do the attack using two threads.
  pthread_create(&pth1, NULL, madviseThread, NULL);
  pthread_create(&pth2, NULL, procselfmemThread, TARGET_CONTENT);

  // Wait for the threads to finish.
  pthread_join(pth1, NULL);
  pthread_join(pth2, NULL);

  return 0;
}
```

Figure 6. The code for Dirty COW exploit.

The "procselfmemThread" method handles resources being used in the runtime. This method calls a file "/proc/self/mem" as read only. This file hold s current memory allocated all the application running in the RAM. The "procselfmemThread" method is shown in Figure 7. Using "lseek" in "/proc/self/mem" file, the memory location of the /etc/passwd of root account is gained. The next command line actually writes over the mapped memory of the file which creates a copy of that file to write to because of the MAP\PRIVATE parameter.

```
void *procselfmemThread(void *arg)
{
    char *str;
    str=(char*)arg;
    int f=open("/proc/self/mem",O_RDWR);
    int i,c=0;
    for(i=0;i<1000000 && !stop;i++) {
        lseek(f,map,SEEK_SET);
        c+=write(f, str, sc_len);
    }
    printf("thread stopped\n");
}
```

Figure 7. The code for Dirty COW exploit: procselfmemThread.

If these two threads would run next to each other there would no change in the root files. To implement the "Dirty COW" attack. These threads are executed over and over again until the edited etc/passwd file is written in the emptied

memory location by the 'madviseThread' method. Thus if the write () operation is performed while the memory is being cleared the kernel overwrites the original file, allowing to edit the write protected file owned by root. Once the overwrite process is successful, the attacker gains the root access.

V. Data Analysis

This paper aims to analyze the affect of Dirty-COW attack in Servers of Bangladesh. As this is an ongoing study, 200 web servers are inspected for initial analysis purpose. Among the 200 servers, 68% are found using various Linux distribution, 30% uses windows operations system and 2% uses various other operations systems as shown in Figure 8.

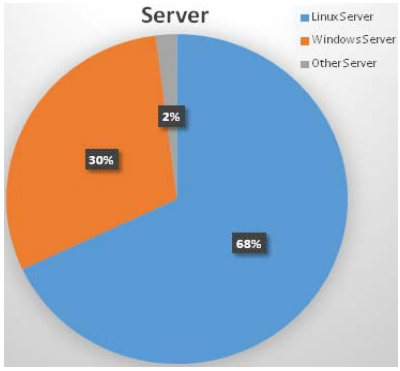

Figure 8. Usage of various operating systems.

The severity effect of the Dirty COW attack partially depends on the category of the host Linux servers. The inspected servers of this study were found using two categories: VPS and Shared host. Both the categories are vulnerable based on the Linux kernel. The effect on the VPS is less as these servers host only one user. The effect of Dirty COW attack is the most on shared host servers as one server can host many web applications. Among the inspected vulnerable servers 54% are shared host while 46% are VPS as shown in Figure 9.

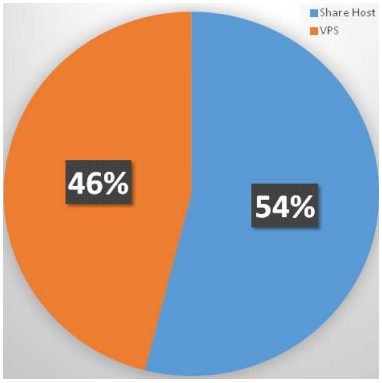

Figure 9. Linux Servers using VPS and shares host.

As all linux kernel are not vulnerable to Dirty COW attack, among the Linux servers inspected about 65% are found vulnerable to the attack as Shown in Figure 10. 30% of the Linux servers are not vulnerable to Dirty COW attack due to using a safe Linux Kernel. The vulnerability of 5% servers were unpredictable.

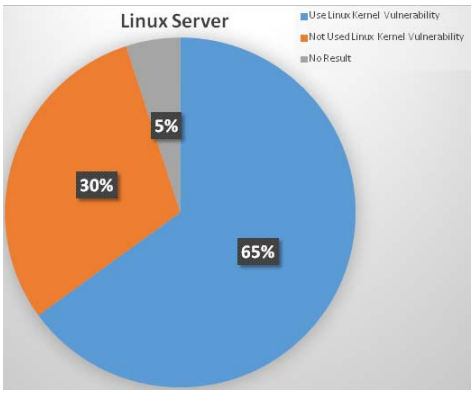

Figure 10. Vulnerability statistics of inspected Linux kernels.

VI. Conclusion

Linux Kernel is known for its open source architecture and best services. It is most popular server side operating system. Though Dirty COW vulnerability existed since 2007, its severity has been recognized in 2016. A fix for this vulnerability has already been released yet the new version of Linux has not been published yet. The growth of online services is rapid in Bangladesh. Unfortunately the security concern related to this growth is not as noteworthy. The extent of Dirty COW vulnerability is large. Number of financial, educational, health care organization, even few government organization are using Dirty COW vulnerable servers to host their web applications. The server administrators as well as the organizations should take this vulnerability into consideration. As per the statistics from our collected data 68\% in Bangladesh are linux server and 65\% of them being vulnerable to Dirty COW attack as of now. This large number of vulnerable applications suggests that, though a fix exists yet the administrators and the organizations are oblivious of the idea of resolving the vulnerability. This negligence is a critical issue with security of web applications in Bangladesh. From government level there should be a protocol about using old version operating systems and slow update of software thus protect the interest of the masses.

References

[1] H. Lan and X. Wang, "Research and Design of Concurrent Web Server on Linux System," International Conference on Computer Science and Service System, Nanjing, pp. 734-737, 2012.

[2] J. Brwere, "Web Server Vulnerabilities and a Defense in Depth Strategy Using the Squid Proxy," online [available] https://www.giac.org/paper/gsec/3729/web-server-vulnerabilities-defense-in-depth-strategy-squid-proxy/105970, [Accessed 12th January 2017].

[3] "Web applications security statistics 2016," online [available] https://www.whitehatsec.com/info/website-stats-report-2016-wp, [Accessed 15th January 2017].

[4] "February 2016 Web Server Survey, Netcraft", [Online]. Available: https://news.netcraft.com/archives/2016/02/22/february-2016-web-server-survey.html, [Accessed: 20- Jan- 2017].

[5] "What Is Root? -- Definition By The Linux Information Project (LINFO)". Linfo.org. N.p., 2017. Web. 18 Jan. 2017.

[6] X. Wu, W. Wang, and S. Jiang. “TotalCOW: Unleash the Power of Copy-On-Write for Thin-provisioned Containers,” In Proceedings of the 6th Asia-Pacifc Workshop on Systems (APSys ’15), New York, NY, USA, pages 15:1–15:7, 2015.

[7] R. D. Kombade, and B. B. Meshram, "CSRF Vulnerabilities and Defensive Techniques" International Journal of Computer Network and Information Security, 4(1), p.31.,2012.

[8] W. Mauerer. “Professional Linux Kernel Architecture,” Wrox Press Ltd., Birmingham, UK, 2008.

[9] O. Rodeh, J. Bacik, and C. Mason, “Btrfs: The linux b-tree filesystem. Trans. Storage,” p. 9(3):9:1–9:32, Aug. 2013.

[10] P. R. Wilson, S. F. Kaplan, and Y. Smaragdakis, “The case for compressed caching in virtual memory systems,” In Proceedings of the Annual Conference on USENIX Annual Technical Conference, (ATEC ’99), pages 8–10, Berkeley, CA, USA, 1999.