**CS392 – Quiz**

Name: **Maheeth Reddy**

Roll No.: **1801CS31**

Date: **18-Apr-2021**

**Ans 1:**

The given program attack.c, consists of a mistake. We are initializing the map variable (void pointer) in the main() function using mmap() function. One of the parameters to mmap() is mentioned as MAP\_SHARED flag. This causes the mapped memory to be shared between two processes. But, this flag is supposed to be MAP\_PRIVATE for the attack to happen successfully.

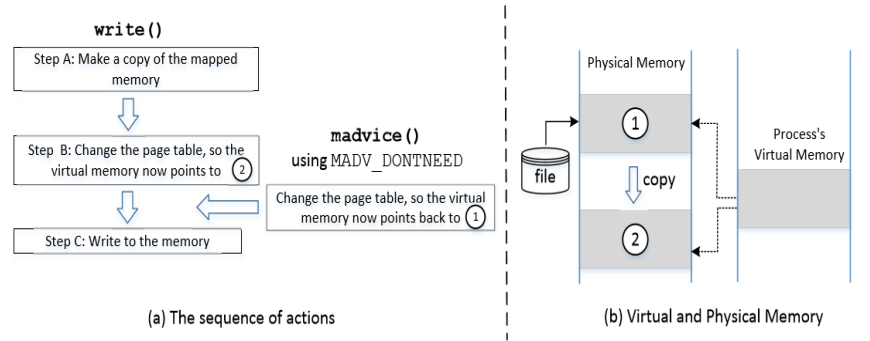
Shankar is trying to do a Dirty-COW attack. The three important steps in this attack are:

(A) Make a copy of the mapped memory

(B) Update the page table, so the virtual memory points to newly created physical memory

(C) Write to the memory.

The principle behind the attack is to create a race condition like TOCTTOU. The steps A,B,C are not atomic in nature: they can be interrupted by other threads which creates a potential race condition leading to Dirty Cow vulnerability.



Picture taken from slides

Consider the following scenario. If madvise() is executed between Steps B and C, Step B will make the virtual memory point to 2. But madvise() will change it back to 1, sort of negating Step B. This causes Step C to modify the physical memory marked by 1, instead of the private copy. Now, the read-only file is modified, which Shankar had no permission to edit. Since this is a COW memory, when the write() system call is invoked, it triggers A,B,C without a double check.

**Ans 3:**

Cross Site Scripting Attack (XSS) occurs when there is a malicious code in the user's input on a particular application. In other words, malicious code to the victim’s browser is injected by the attacker via the target website.

There are two types of XSS attacks:

1) Non-Persistent XSS Attack

Attacker puts JavaScript code in the input. So when the input is reflected back, the JavaScript code will be injected into the web page from the website.

2) Persistent XSS Attack

In this case, the attacker stores malicious data (code) in target websites servers. And when other users access this data, they shall be affected.

In either of the cases, the attacker looks for an attack surface where they can input the malicious code. Therefore it is highly essential to sanitize the input for code.

One way to do this is to identify the <script> tags and remove them. But there are other ways to inject code other than using the <script> tag. It is NOT THE ONLY MEASURE, as there are several encoding methods which would convert HTML Markups into non-malicious strings and various open source libraries to parse the Input from JavaScript Codes (Like jsoup)