Short Report

Outline

Initially, all the pages (pte entries) in all the vm areas in the process are made "faulty", by putting the present bit 0 and protected bit 1. In this way, even if the page is present in the memory, user programs cannot access it and hence a page fault will occur.

The kernel source file where the page fault handling occurs, is "/arch/x86/mm/fault.c". Inside the function __do_page_fault(), a kernel hook was added before the handle_mm_fault() function call where the task struct and page address is passed.

In the hook function implemented in the kernel module, the task_struct is checked whether it is the same process as the command line argument. If it is the same, then the page table entry (pte) is extracted and checked for flags. If it is found to be "not present" and protected", it might be done by the lkm. To verify this, the address is checked against the linked list of "modified pages". If found to be present, wss_count is incremented and the pte flags are set to "present" and "not protected".

PERIODICALLY, all the page_addresses in the linked list are made "present" and "not protected", the linked list is freed, wss_count is LOGGED, and the whole algorithm starts again.

High Level Algorithm

1. Init wss_count = 0

2. fault_all_pages():

    for each vm_area associated with the pid:

      for each page in the vm_area:

        To ensure a page fault on access,

        set the page flags such that it is

          NOT present

          Protected

        Add the page address to a linked list for checks

3. Point the kernel hook to the function my_fault_handler

  The function algorithm is given below

4. Repeat the following steps "times" number of times

   times is obtained as a command line argument

---

   a. LOG the wss_count

   b. set all pages pte in the linked list with flags

      <present> and <NOT protected>

   c. free the linked list

   d. call fault_all_pages() [given as step 2]

   e. reset wss_count to 0

---

5. set the kernel hook back to NULL

6. set all pages pte in the linked list with flags

   <present> and <NOT protected>

7. Free the linked list

Kernel Hook Functionality

my_fault_handler(task_struct *tsk, unsigned long page_addr):

   CHECKING WHETHER IT IS A PAGE_FAULT GENERATED BY THE LKM

   if the tsk->pid is the target pid:

      get the pte of the page address "page_addr"

         if pte is NOT present and protected:

            if "page_addr" comes within the linked list

               set the pte flags as present and NOT protected

               increment wss_count

Experiment to test

1. Compile the program "tester.c"

```
gcc -o tester tester.c
```

2. Run the program with command line argument 10

   The program will try to periodically access a random number of pages

   with 10 second time gaps

   for 10 times

   It will print out its PID.

3. Insert the kernel module wss.ko, with pid as PID printed in step 2

   and times as 10

```
sudo insmod wss.ko pid=<PID> times=10
```

4. Compare the output of the "tester" program with the syslogd output

   It was seen that the outputs were corresponding.

   The change in adjacent "number of pages accessed" (program output)

   is EXACTLY EQUAL TO

   the change in adjacent "estimated WSS" (lkm output).

   Conclusion: WSS estimation is correct.

Test OUTPUT

The output from syslogd, the tester program, **the differences of each output with its previous output – The number of new pages accessed** – is considered.

Syslogd	**Difference**	**Tester**	**Difference**	**Test_Result**
Module started		Tester; Process ID: 6518		
Wss = 31	-	Accessing 5 pages	-	Success
Wss = 34	+3	Accessing 8 pages	+3	Success
Wss = 34	+0	Accessing 8 pages	+0	Success
Wss = 31	-3	Accessing 5 pages	-3	Success
Wss = 29	-2	Accessing 3 pages	-2	Success

Syslogd	Difference	Tester	Difference	Test_Result
Wss = 35	+6	Accessing 9 pages	+6	Success
Wss = 26	-9	Accessing 0 pages	-9	Success
Wss = 33	+7	Accessing 7 pages	+7	Success
Wss = 29	-4	Accessing 3 pages	-4	Success
Wss = 35	+6	Accessing 9 pages	+6	Success
Wss = 34	-1	Accessing 8 pages	-1	Success

Logging completed. Remove module to print log

The wss was accurately estimated.