作者: 王駿逸 Simon Wang

作業環境: Kali Linux 2016 64 bit + GNU gdb (Debian 7.11.1-2) 7.11.1

日期: 2017年8月

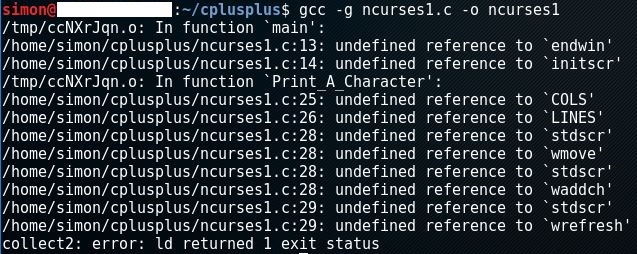
在看到 “Linux kernel live debugging, how it's done and what tools are used?” 和 “Using kgdb, kdb and the kernel debugger internals” 這兩篇文章之後，我們開始意識到、不管是採用 QEMU + GDB 或是 KDB 的方式來 Debugging Live 的 Linux Kernel 、我們都不太可能就只在一個終端機(Terminal ，tty)內就完成它，一定得要兩台機器連接、或是 “一台終端機執行虛擬機 另一台終端機再用 GDB 來連結它” 的方式。

所以有可能我們之前耗費這麼多時間跟步驟、卻還是卡住的 QEMU -s -S 選項啟動時被 Audio Driver 卡住的問題，其實並不是問題，而是它已經正常運作了、只是 Kernel 停止下來、要等待我們用另一台終端機來跟它連線啟動它。

藉著 “Using GNU's GDB Debugger Debugging Ncurses Programs” 這篇文章，我們來練習在 GDB 當中操作 tty (終端機) 參數。這一篇的作用跟用意是: ncurses1.c 是一個有錯誤的 Ncurses 圖形處理程式、由於 Debugging 時所出現的繁雜且混雜的訊息，我們必須開始學習當這個程式在作圖形處理的 I/O 時(也就是在目前的終端機 tty 之下)、我們要把 GDB Debugging 的 I/O 分流到另一個終端機去。

由於 Ncurses 圖形處理並不是我們要探討的重點、我們的重點還是在 GDB ，所以我們對於這個 ncurses1.c 的程式碼有個初步的了解即可。

在完成了 “Using GNU's GDB Debugger Debugging Ncurses Programs” 文章中的程式碼之後，把它來 Compile :



看來 Ncurses 並不是一個預設的 Library(函式庫) ，既然它不是我們的主軸、那我們還是改用比較方便的程式碼好了。我們的比較簡單的程式碼:



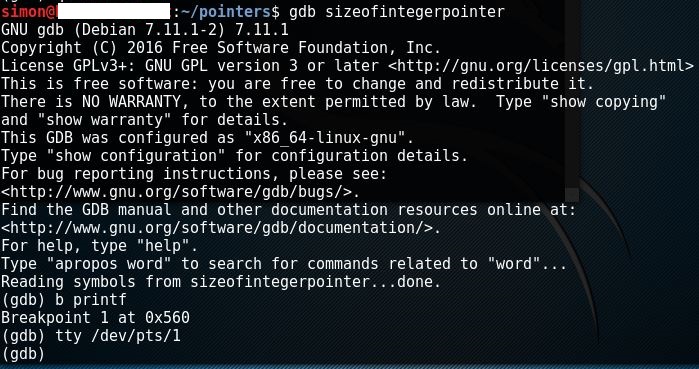
使用 gcc -g sizeofintegerpointer.c -o sizeofintegerpointer 來 Compile 之後，我們要先開啟兩個終端機 (tty) :

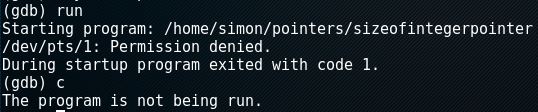
C:\Users\kdbot\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Using GNU s GDB Debugger Debugging Ncurses Programs 3 我們原本所在的終端機.jpg

另一個我們所新開的終端機:

C:\Users\kdbot\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Using GNU s GDB Debugger Debugging Ncurses Programs 4 另一個我們所新開的終端機.jpg

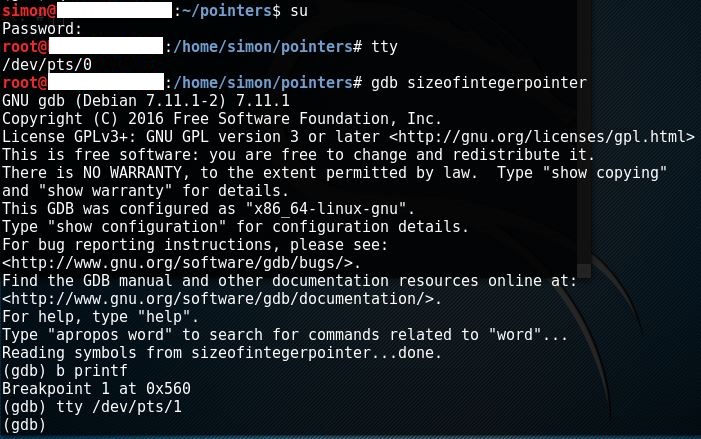
再來， GDB 的 tty 指令可以把 GDB 的運作輸出結果導向到我們所指定的終端機去，因此我們先在 /dev/pts/0 這個終端機上設定 GDB 的中斷點、並設定 tty 指令來指定到新的 /dev/pts/1 終端機上:





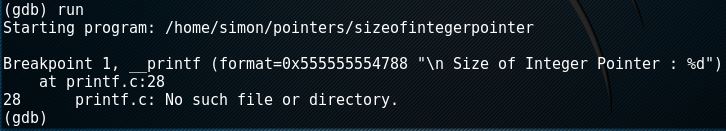
結果出現錯誤: /dev/pts/1: Permission denied. 無法同步到 pts/1 終端機、在 /dev/pts/1 這個終端機上也沒有顯示任何訊息。

思考一下，既然出現的錯誤訊息是 Permission denied 、而我們在這兩個終端機所使用的使用者都是 simon(一般權限的使用者，也不具有 sudo 權限)，所以如果我們把這兩個終端機都提升權限為 root 的話、看看會如何?





現在兩個終端機的 User 都是 root 。



在 /dev/pts/0 終端機上。

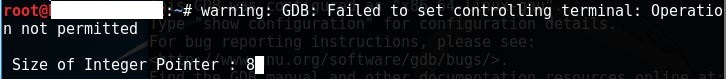
C:\Users\kdbot\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Using GNU s GDB Debugger Debugging Ncurses Programs 7 提升兩個終端機的使用者的權限都為 root 試試看 4 在 pts 1 終端機上.jpg

在 /dev/pts/1 終端機上。

我們再讓 /dev/pts/0 的 GDB 單步執行與繼續執行下去看看:



在 /dev/pts/0 終端機上。



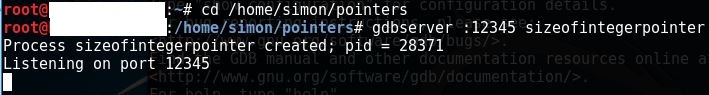
在 /dev/pts/1 終端機上。雖然有出現 warning 訊息、但還是有顯示出我們要的結果，所以我們的試驗算是成功了。

我們還有另一種方法，那就是使用 gdbserver :

同樣的兩邊的終端機使用者都是 root :

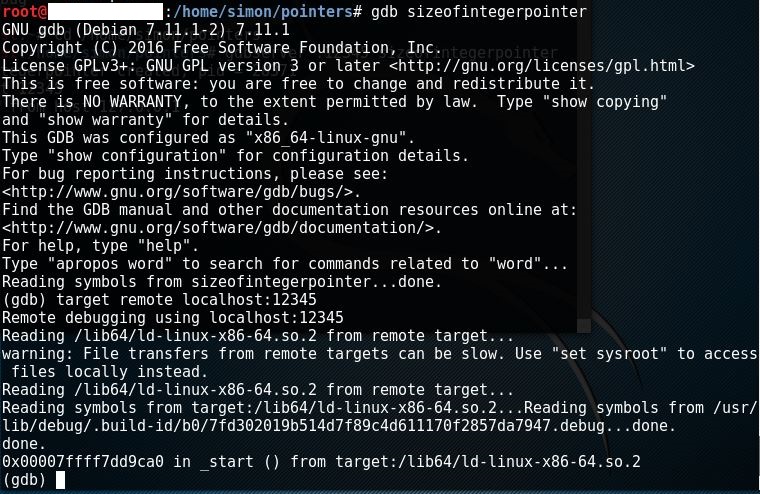


找不到這個 sizeofintegerpointer 要被 Debug 的程式，所以我們就切換到 sizeofintegerpointer 它的所在目錄下:

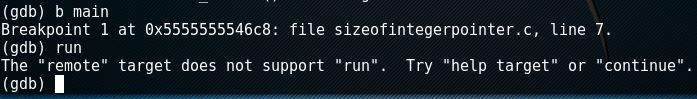


然後在另一個終端機的相同目錄下:

輸入指令: target remote localhost:12345

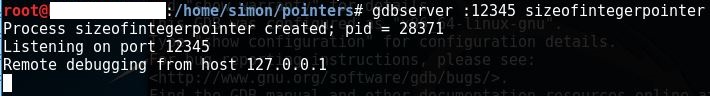


接著:

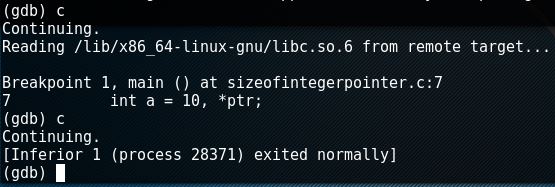


注意到它 remote target 並不支援 run 這個指令。

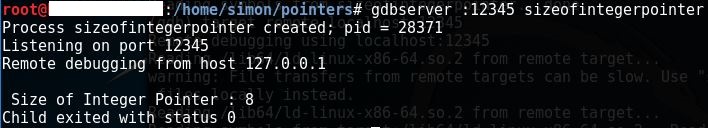
此時在 gdbserver 這個終端機就出現了:



再回到 GDB 讓它繼續執行下去:



到此 gdbserver 的終端機就有顯示出我們要的結果了:



所以就算是大功告成了。

Reference:

1. GDB - Display output of target application in a separate window:

<https://stackoverflow.com/questions/8963208/gdb-display-output-of-target-application-in-a-separate-window>

1. Using GNU's GDB Debugger Debugging Ncurses Programs:

<http://dirac.org/linux/gdb/07-Debugging_Ncurses_Programs.php>

1. Program 1: Size of Integer Pointer:

<http://www.c4learn.com/c-programming/c-pointer-variable-memory-required/>