OHTS – IE4012

Offensive Hacking Tactical and Strategic

M.A.T.Kavindi – IT17403948

Assignment

Exploit development of CVE-2018-5333

# 1.Discription about this CVE

CVE Number: CVE-2018-5333

Severity: Medium

Published: January 11, 2018

Description: In the Linux kernel through 4.14.13, the rds\_cmsg\_atomic function in net/rds/rdma.c mishandles cases where page pinning fails or an invalid address is supplied, leading to an rds\_atomic\_free\_op NULL pointer dereference

List of vulnerable kernals:

|  |  |  |  |
| --- | --- | --- | --- |
| Linux kernel 4.14.13 + EnGarde Secure Linux 1.0.1 + Immunix Immunix OS 7+ + MandrakeSoft Single Network Firewall 7.2 + Mandriva Linux Mandrake 8.1 + Mandriva Linux Mandrake 8.0 ppc + Mandriva Linux Mandrake 8.0 + SuSE Linux 7.0 + SuSE Linux 6.4 + SuSE Linux 6.3 + Trustix Secure Linux 1.5 Linux kernel 4.14.11 Linux kernel 4.14.10 Linux kernel 4.14.6 Linux kernel 4.14.5 Linux kernel 4.14.1 Linux kernel 4.13.11 Linux kernel 4.13.10  Linux kernel 4.13.9  Linux kernel 4.13.8  Linux kernel 4.13.7  Linux kernel 4.13.6  Linux kernel 4.13.4  Linux kernel 4.13.3  Linux kernel 4.12.9 | Linux kernel 4.12.3  Linux kernel 4.12.2  Linux kernel 4.11.9  Linux kernel 4.11.5  Linux kernel 4.11.4  Linux kernel 4.11.3  Linux kernel 4.11.2  Linux kernel 4.11.1  Linux kernel 4.11  Linux kernel 4.10.15  Linux kernel 4.10.13  Linux kernel 4.10.12  Linux kernel 4.10.10  Linux kernel 4.10.6  Linux kernel 4.10.4  Linux kernel 4.10  Linux kernel 4.1.47  Linux kernel 4.1.4  Linux kernel 4.1.1  Linux kernel 4.0.6  Linux kernel 3.19.3  Linux kernel 3.18.22  Linux kernel 3.18.17  Linux kernel 3.18.11  Linux kernel 3.18.8  Linux kernel 3.18.7  Linux kernel 3.18.3  Linux kernel 3.18.2  Linux kernel 3.18.1  Linux kernel 3.17.4  Linux kernel 3.17.2  Linux kernel 3.16.7 | Linux kernel 3.16.2  Linux kernel 3.16.1  Linux kernel 3.15.10  Linux kernel 3.15.5  Linux kernel 3.15.2  Linux kernel 3.14.54  Linux kernel 3.14.45  Linux kernel 3.14.37  Linux kernel 3.14.4  Linux kernel 3.14.3  Linux kernel 3.14.2  Linux kernel 3.13.11  Linux kernel 3.13.9  Linux kernel 3.13.3  Linux kernel 3.13.1  Linux kernel 3.12.49  Linux kernel 3.12.48  Linux kernel 3.12.44  Linux kernel 3.12.40  Linux kernel 3.12.21  Linux kernel 3.12.18  Linux kernel 3.12.17  Linux kernel 3.12.16  Linux kernel 3.12.11  Linux kernel 3.12.7  Linux kernel 3.12.4  Linux kernel 3.12.3  Linux kernel 3.12.2  Linux kernel 3.11.3  Linux kernel 3.10.90  Linux kernel 3.10.81  Linux kernel 3.10.73 | Linux kernel 3.10.73  Linux kernel 3.10.45  Linux kernel 3.10.41  Linux kernel 3.10.38  Linux kernel 3.10.37  Linux kernel 3.10.36  Linux kernel 3.10.30  Linux kernel 3.10.27  Linux kernel 3.10.26  Linux kernel 3.10.23  Linux kernel 3.10.22  Linux kernel 3.10.21  Linux kernel 3.10.14  Linux kernel 3.10.10  Linux kernel 3.10.9  Linux kernel 3.10.7  Linux kernel 3.10  Linux kernel 3.8.9  Linux kernel 3.8.6  Linux kernel 3.8.5  Linux kernel 3.8.4  Linux kernel 3.8.2  Linux kernel 3.8.1  Linux kernel 3.7.10  Linux kernel 3.7.9  Linux kernel 3.7.8 |

Updated: Jan 03 2018

# 2.About target OS

## 2.1. Target Kernel 1: Ubuntu 16.04 kernels 4.4.0 <= 4.4.0-116

There are two types of images in this, one for the desktop and other is for the server.

Desktop image will give a chance to user to try Ubuntu without changing he’s computer, and at your option to install it permanently later. This type of image is what most people will want to use. You will need at least 384MiB of RAM to install from this image. This contains,

* 64-bit PC (AMD64) desktop image
* 32-bit PC (i386) desktop image

The server install image allows you to install Ubuntu permanently on a computer for use as a server. It will not install a graphical user interface.

* 64-bit PC (AMD64) server install image
* 32-bit PC (i386) server install image

## 2.2. Target Kernel 2: Ubuntu 16.04 kernels 4.8.0 <= 4.8.0-54

There are two types of images in this also mentioned as above.

# 3. About exploit Code

## 3.1.

Contributors: bcoles, nstarke

Type: Kernel exploit

Language: c

Requirements: Since the rds kernel module is not loaded by default on Ubuntu and is blacklisted

in /etc/modprobe.d/blacklist-rare-network.conf to prevent autoloading, following steps may require running the code,

* Install "linux-image-extra-$(uname -r)-generic".

Command: sudo apt install "linux-image-extra-$(uname -r)-generic"

* Load insmod

Command: sudo insmod "/lib/modules/$(uname -r)/kernel/net/rds/rds.ko"

This is a recreation of the null pointer deference. This exploit adds offsets for additional kernels, and introduces some additional features, such as KASLR bypasses and system checks, including:

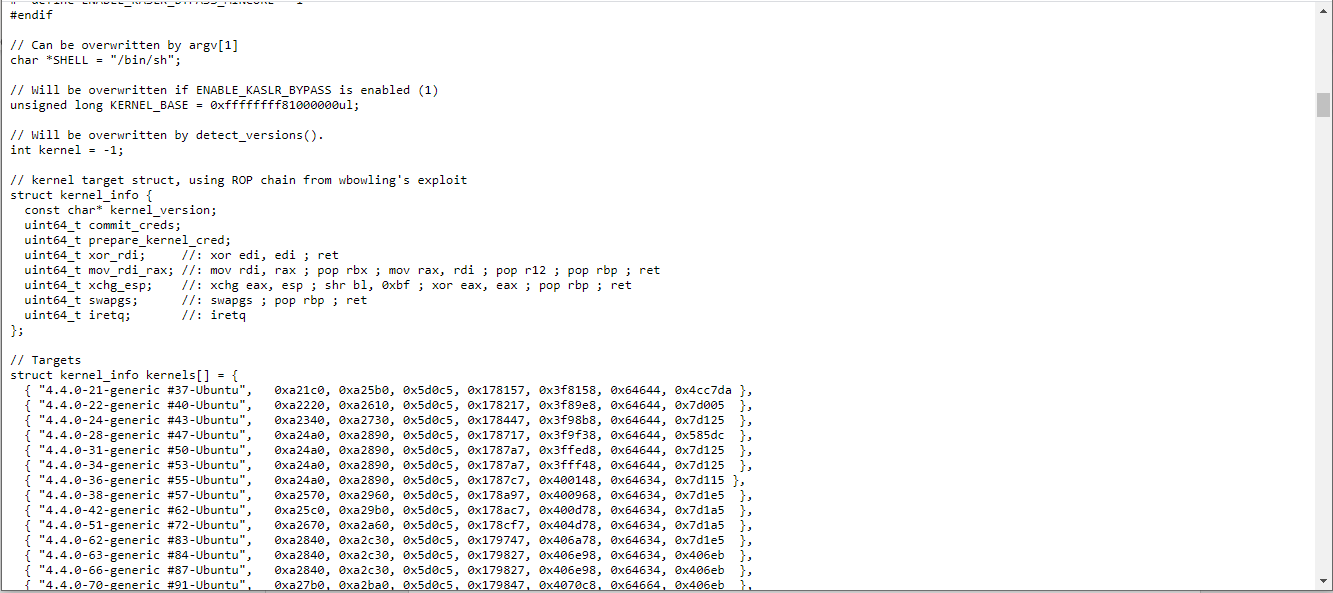
* check if system supports SMAP
* check if system supports RDS sockets
* Jann Horn's mincore KASLR bypass via heap page disclosure (CVE-2017-16994)
* spender's /proc/kallsyms KASLR bypass (requires kernel.kptr\_restrict=0)
* xairy's syslog KASLR bypass (requires kernel.dmesg\_restrict=0)
* lizzie's perf\_event\_open KASLR bypass (requires kernel.perf\_event\_paranoid<2)

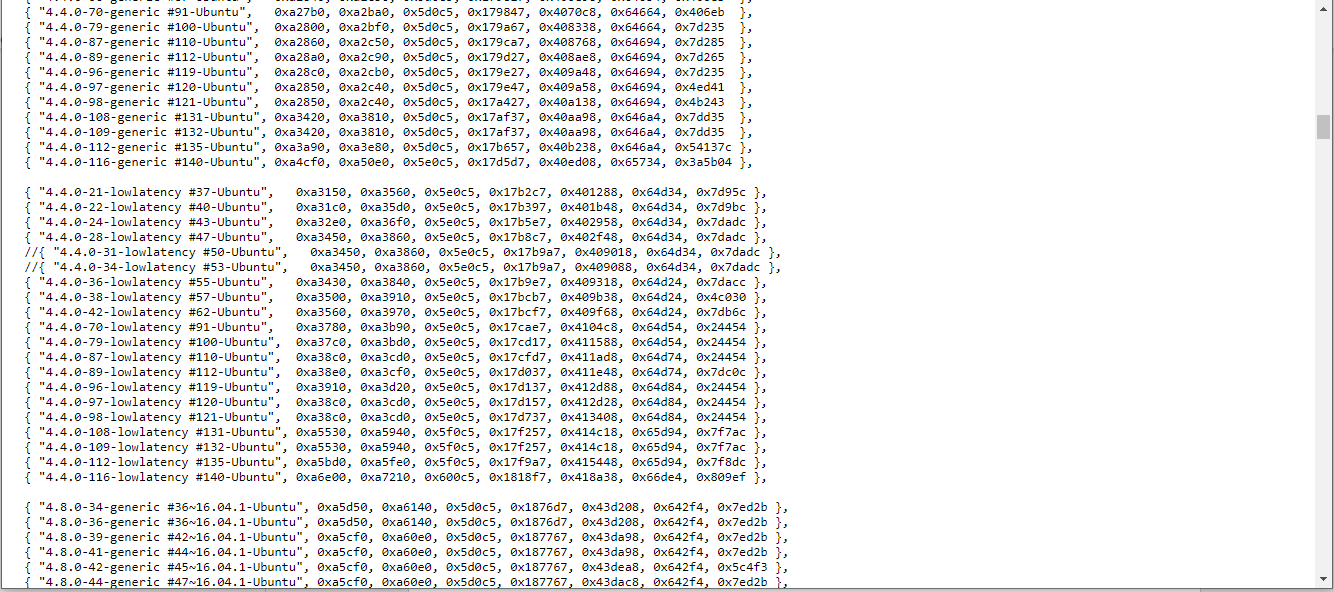
## 3.2. Libraries of the code and their purposes

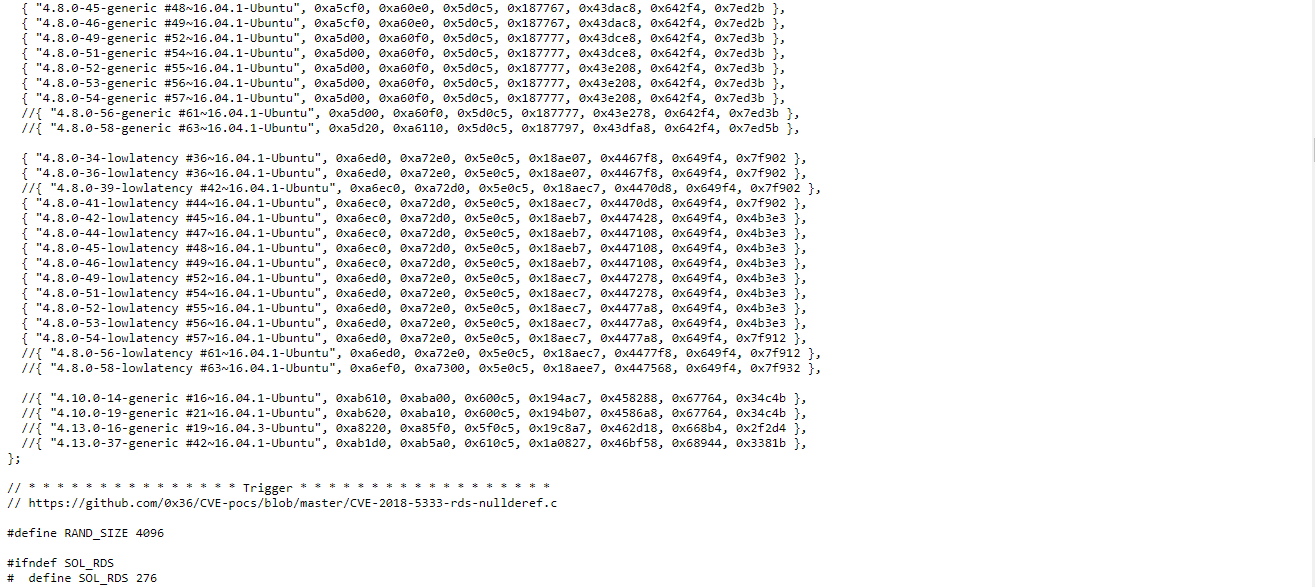
|  |  |  |
| --- | --- | --- |
|  | Library | Purpose |
| 01 | fcntl.h | this containing construct that refer to file control such as opening a file, retrieving and changing the permissions of file, locking a file for edit, etc. |
| 02 | signal.h | This defines a variable type sig\_atomic\_t, two function calls, and several macros to handle different signals reported during a program's execution. |
| 03 | stdio.h | defines three variable types, several macros, and various functions for performing input and output. |
| 04 | stdlib.h | defines four variable types, several macros, and various functions for performing general functions. |
| 05 | string.h | defines one variable type, one macro, and various functions for manipulating arrays of characters. |
| 06 | stdint.h | declare sets of integer types having specified widths and shall define corresponding sets of macros. It shall also define macros that specify limits of integer types corresponding to types defined in other standard headers. |
| 07 | unistd.h | provides access to the POSIX operating system API. It is defined by the POSIX.1 standard, the base of the Single Unix Specification, and should therefore be available in any conforming operating system/compiler |
| 08 | linux/perf\_event.h | Check if event can be used for aux\_output purposes for. |
| 09 | netinet/in.h | define the sockaddr\_in structure that includes at least the following members: ... The sockaddr\_in structure is used to store addresses for the Internet address family. Values of this type shall be cast by applications to struct sockaddr for use with socket functions. |
| 09 | sys/ioctl.h | used to control the Inputs and outputs i.e. to interact with the hardware and the user. |
| 10 | sys/klog.h | The GNU C Library is free software; you can redistribute it and/or modify it under the ... of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE |
| 11 | sys/mman.h | define the structure posix\_typed\_mem\_info, which includes at least the following member: size\_t posix\_tmi\_length Maximum length which may be allocated from a typed memory object. The following shall be declared as functions and may also be defined as macros. |
| 12 | sys/socket.h | defines the following macros to gain access to the data arrays in the ancillary data associated with a message header: CMSG\_DATA(cmsg) If the argument is a pointer to a cmsghdr structure, this macro returns an unsigned character pointer to the data array associated with the cmsghdr structure. |
| 13 | sys/stat.h | header in the C POSIX library for the C programming language that contains constructs that facilitate getting information about files attributes. |
| 14 | sys/syscall.h | saves CPU registers before making the system call, restores the registers upon return from the system call, and stores any error code returned by the system call in errno(3) if an error occurs. Symbolic constants for system call numbers can be found in the header file |
| 15 | sys/types.h | include definitions for at least the following types: blkcnt\_t Used for file block counts. blksize\_t Used for block sizes. clock\_t Used for system times in clock ticks or CLOCKS\_PER\_SEC; see <time. ... clockid\_t Used for clock ID type in the clock and timer functions. |
| 16 | sys/utsname.h | define the structure utsname which shall include at least the following members: char sysname[] Name of this implementation of the operating system. char nodename[] Name of this node within the communications network to which this node is attached |

## 3.3. Code

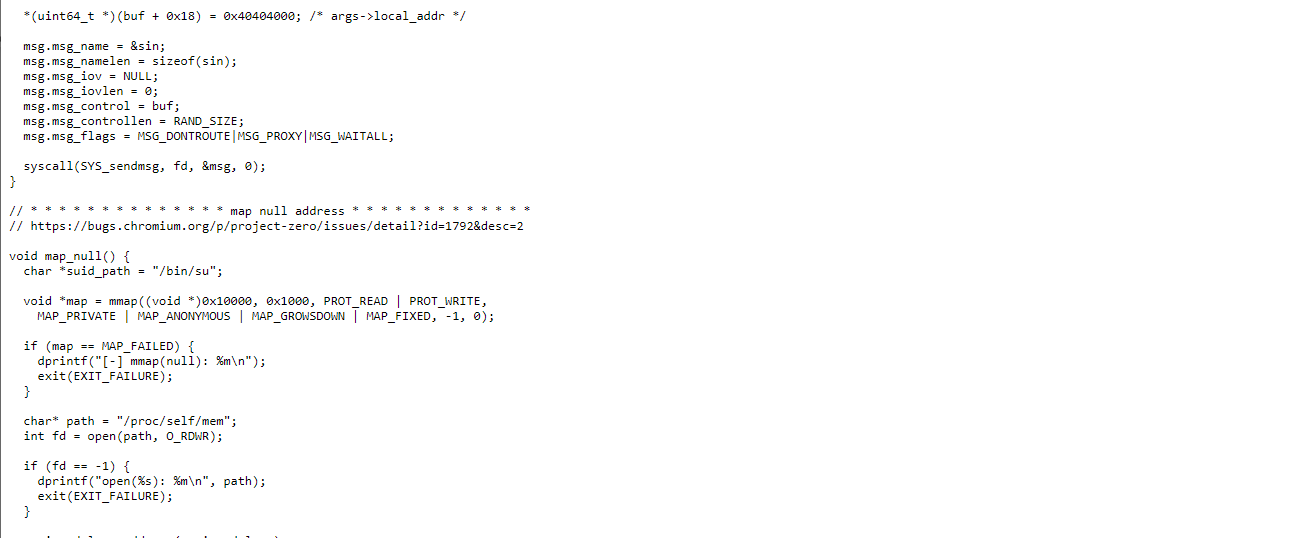


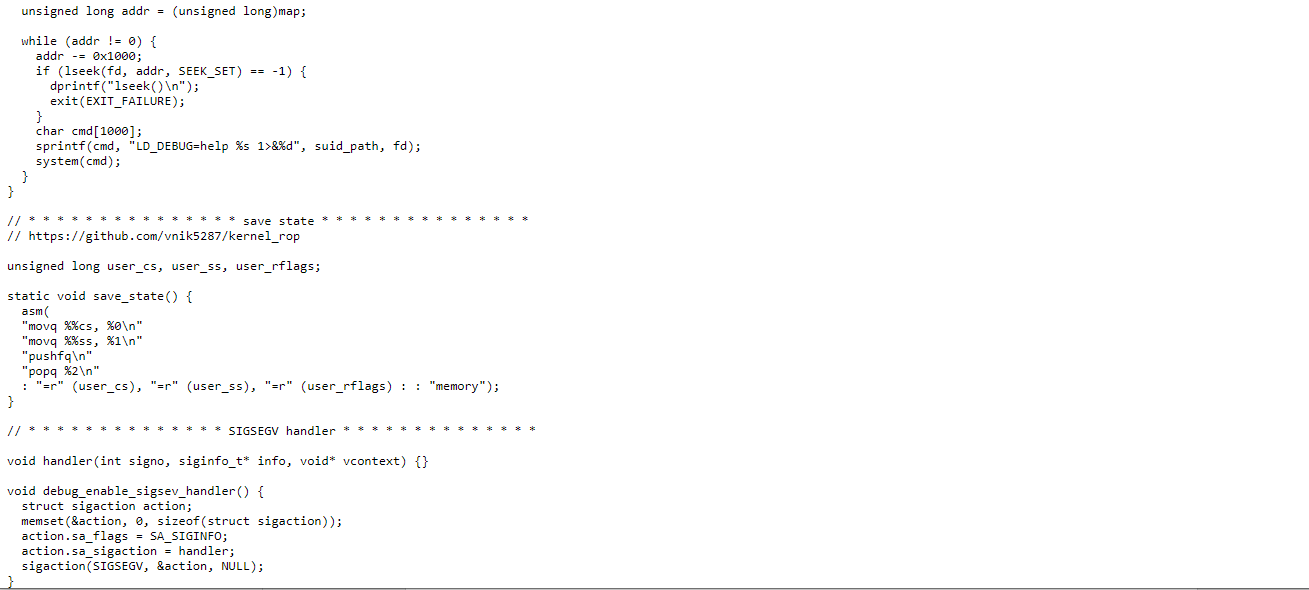


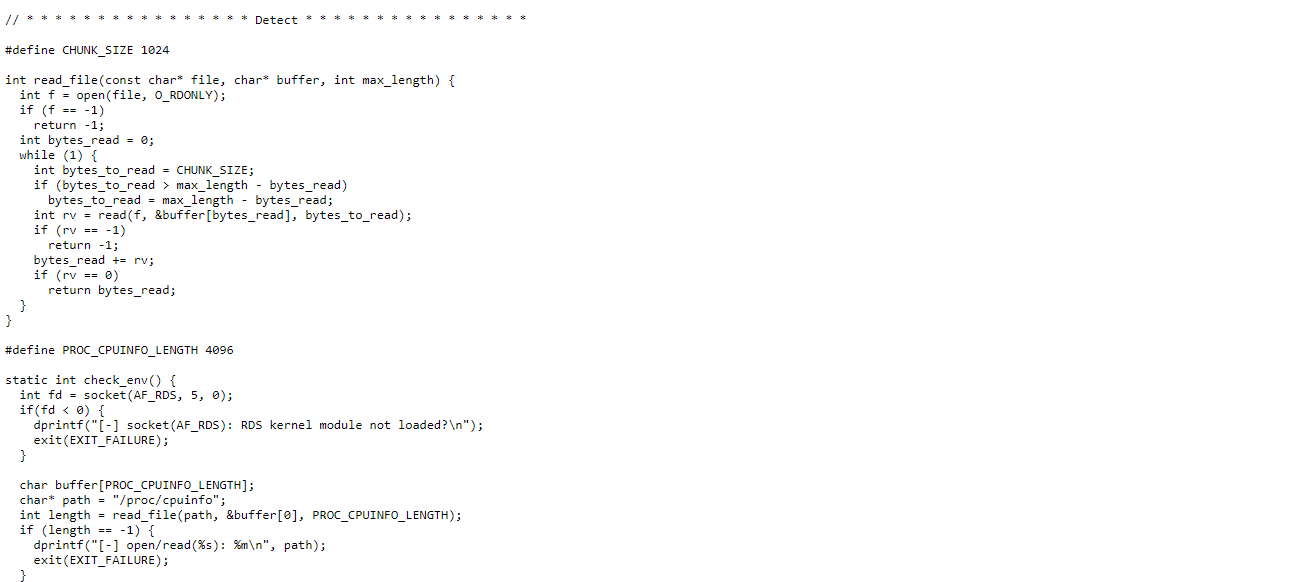




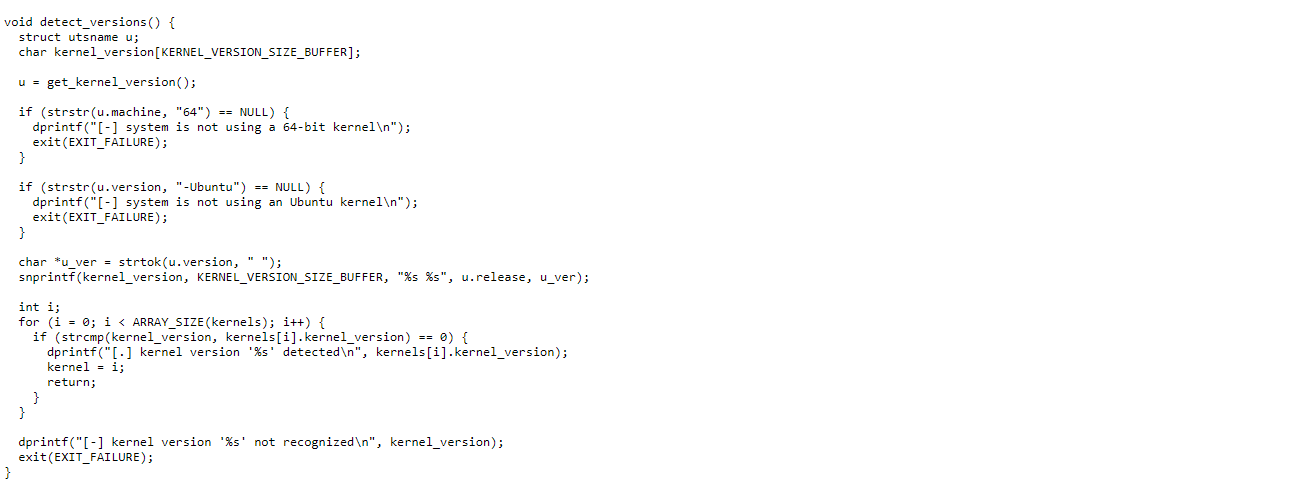


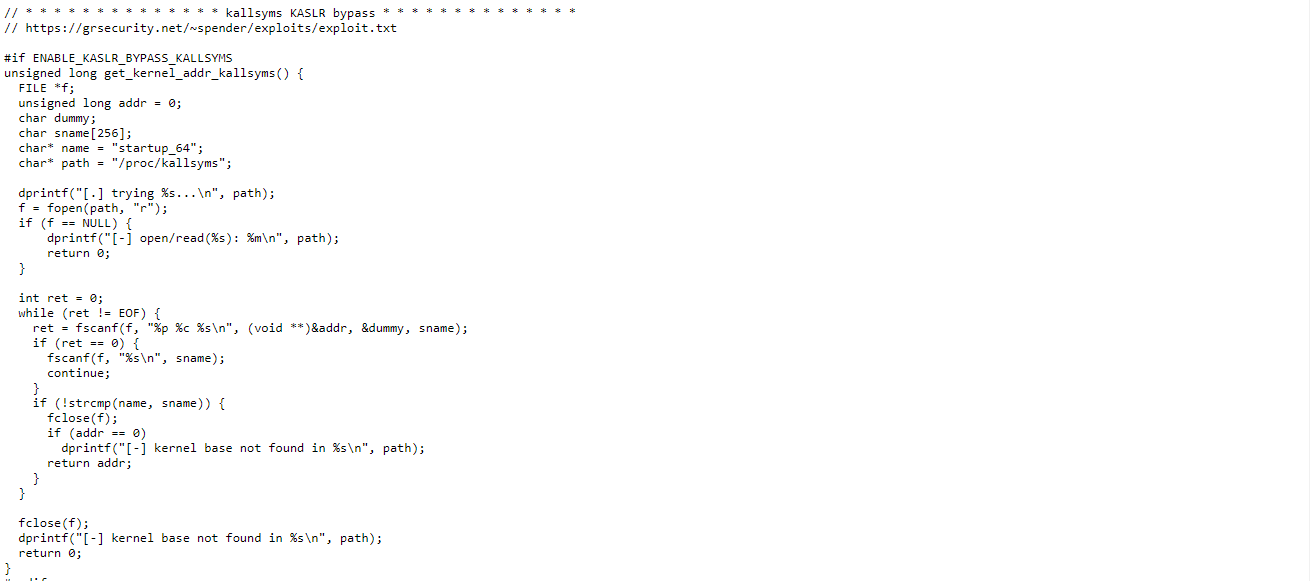


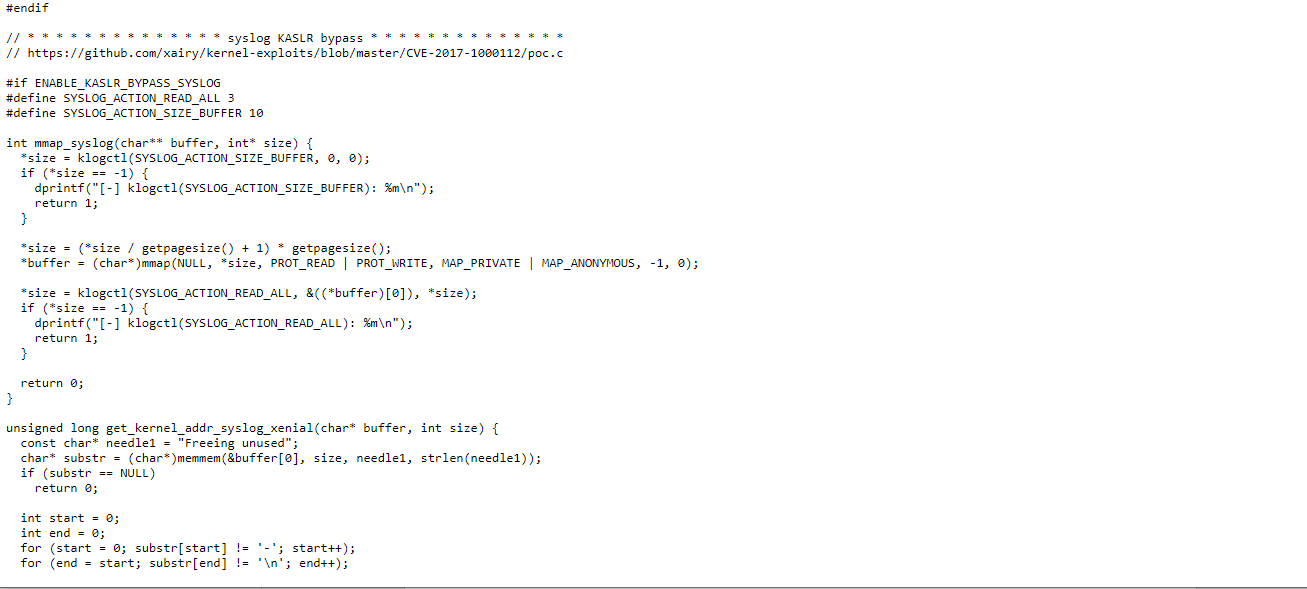


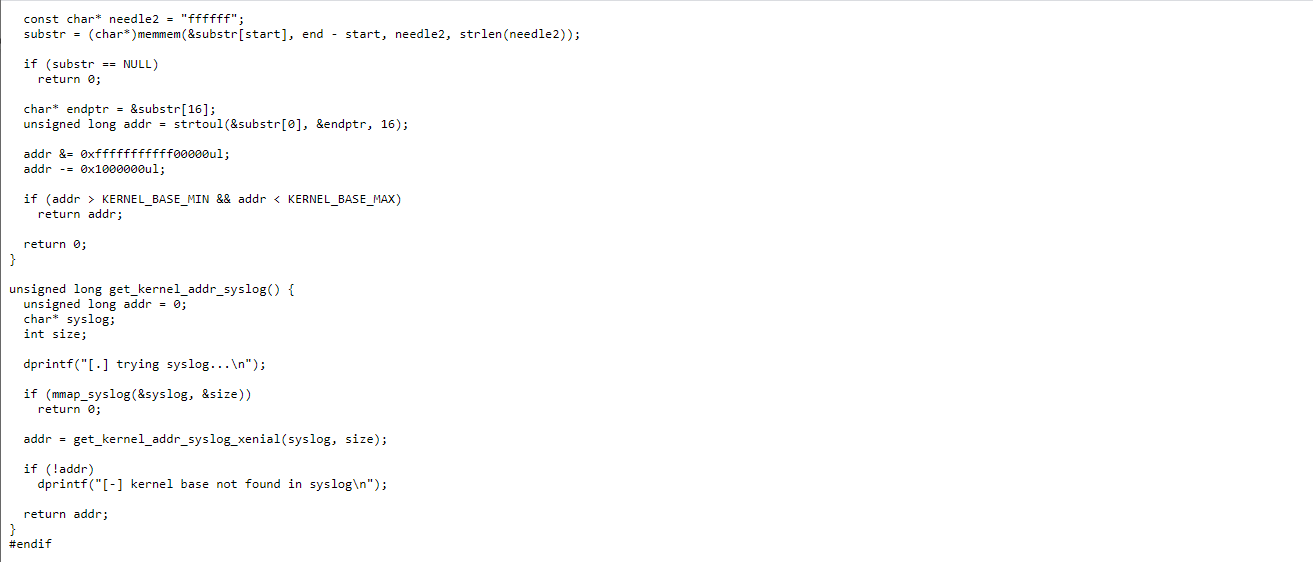


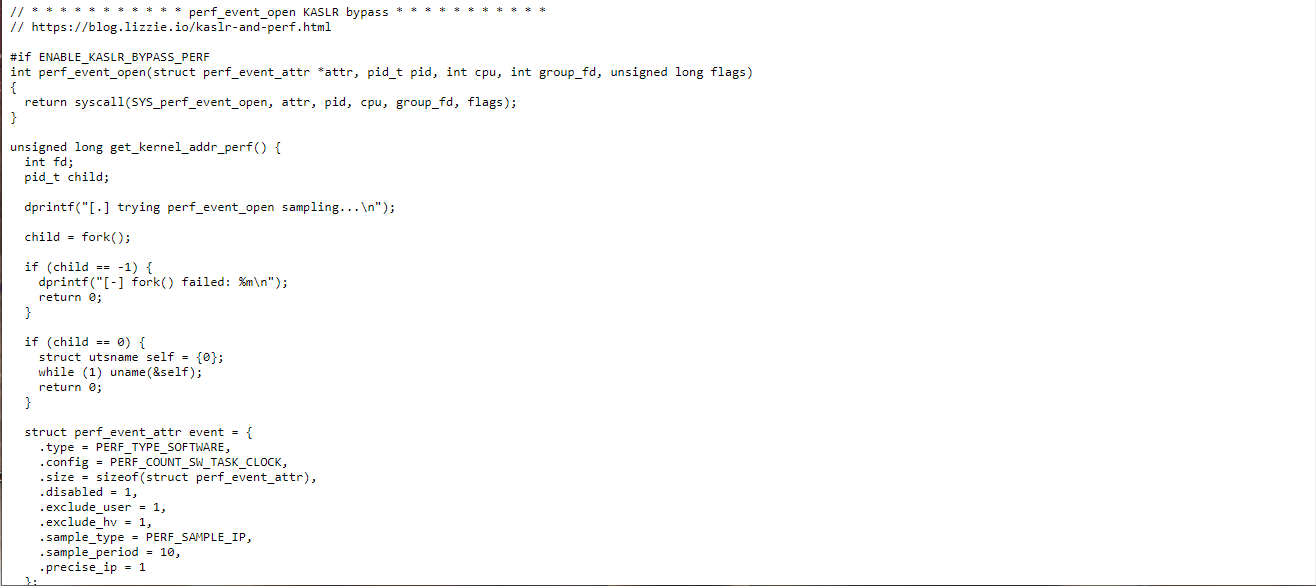


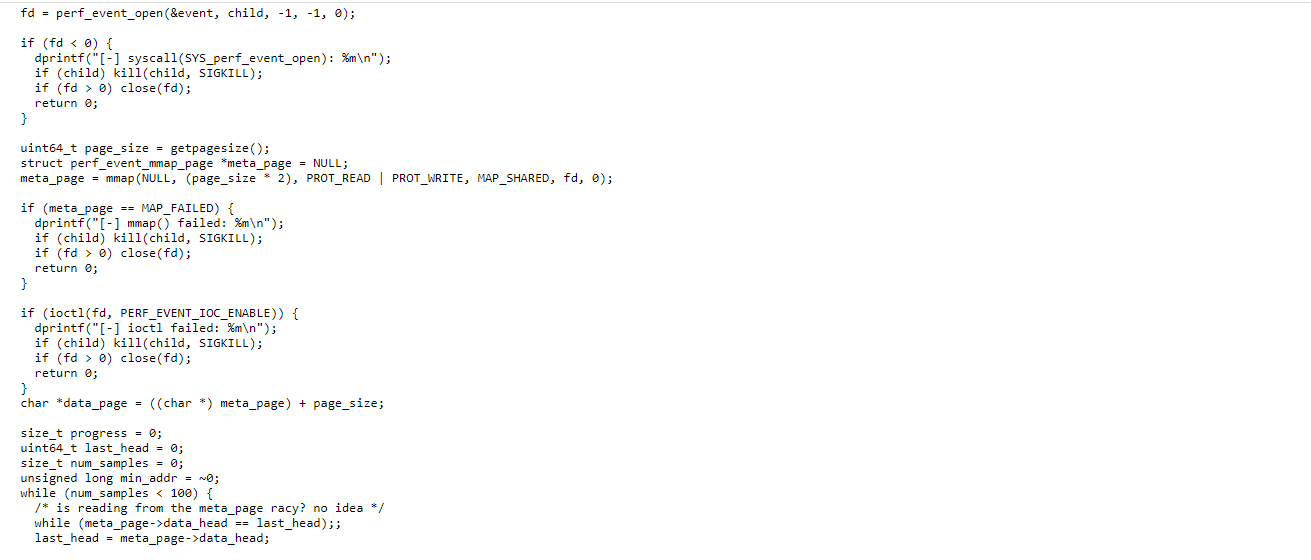


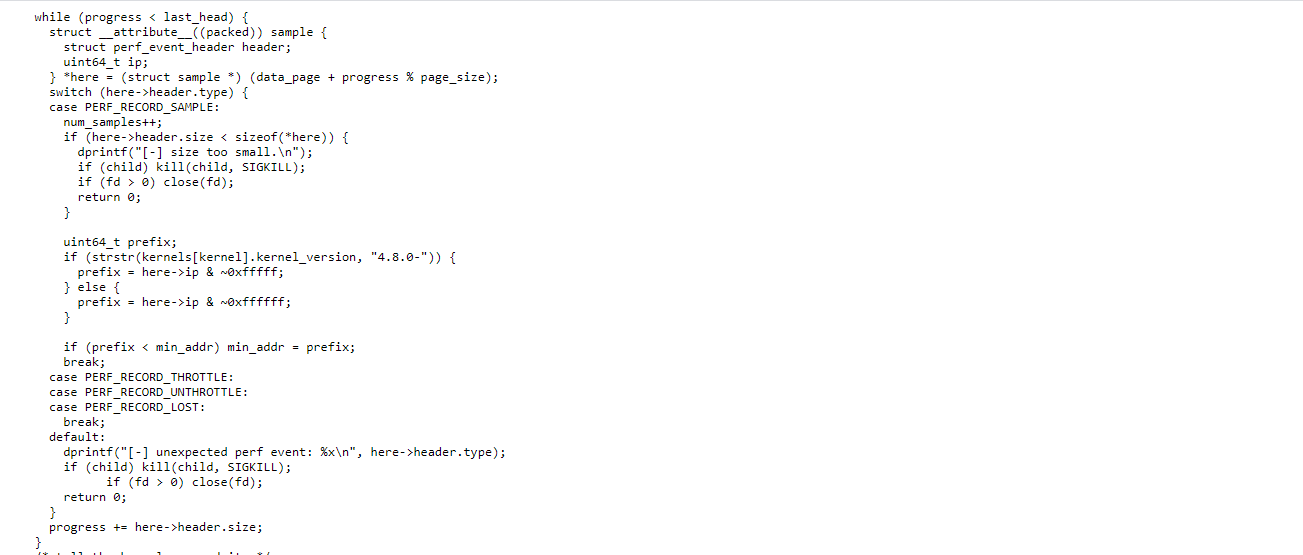


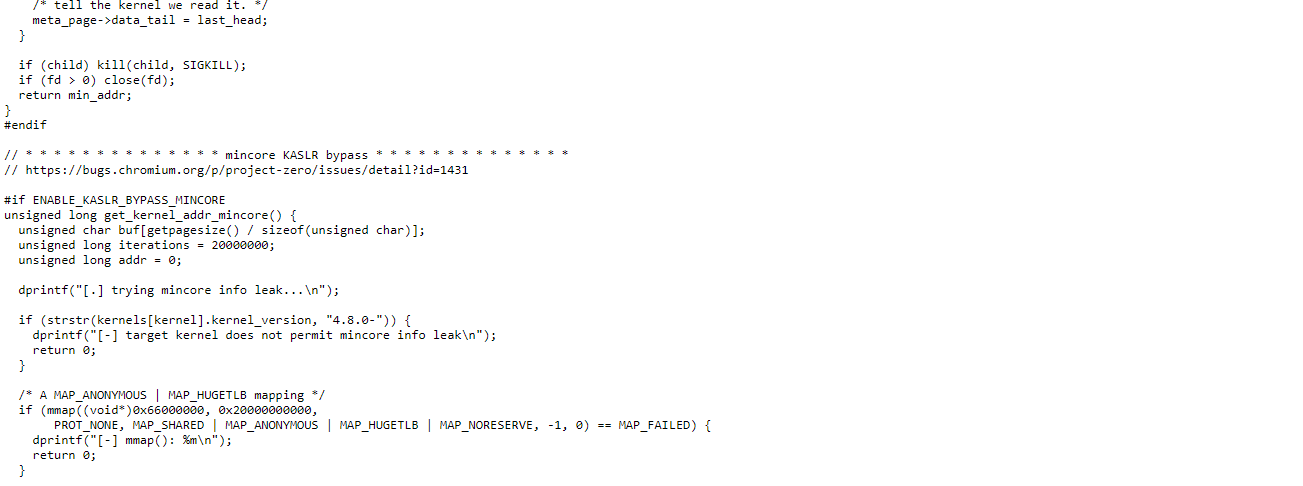


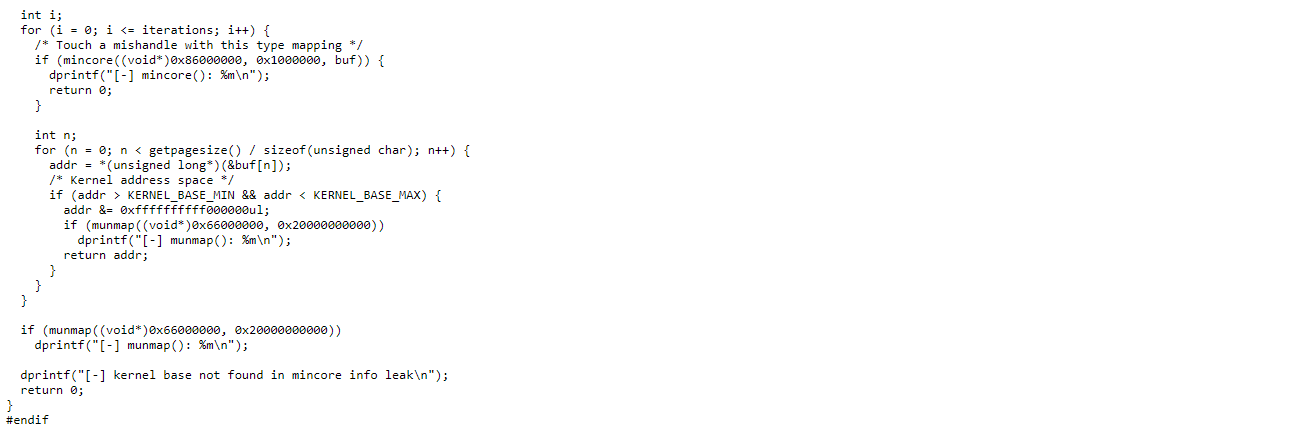


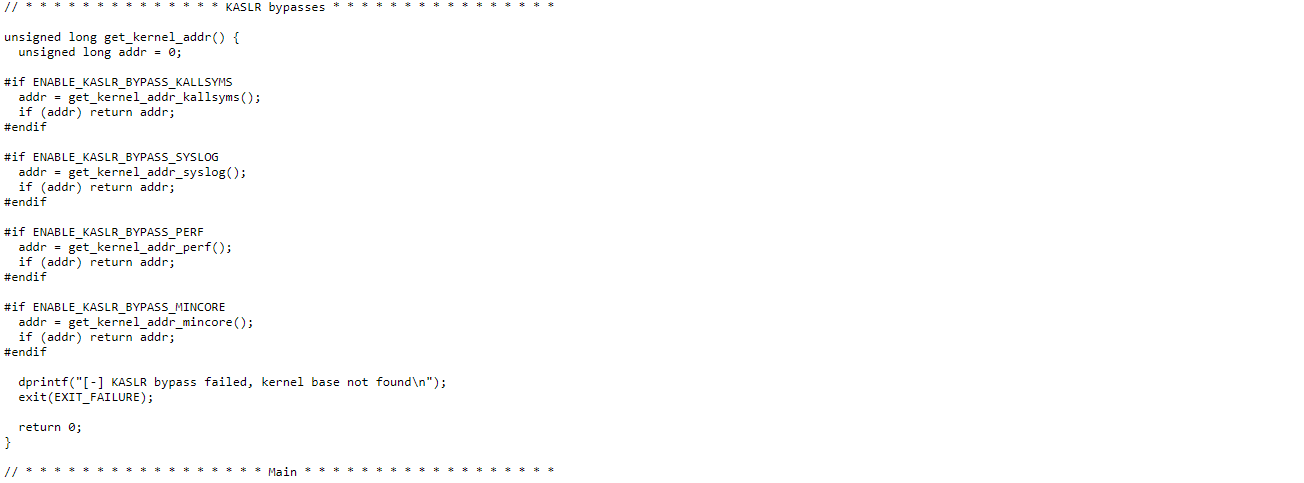


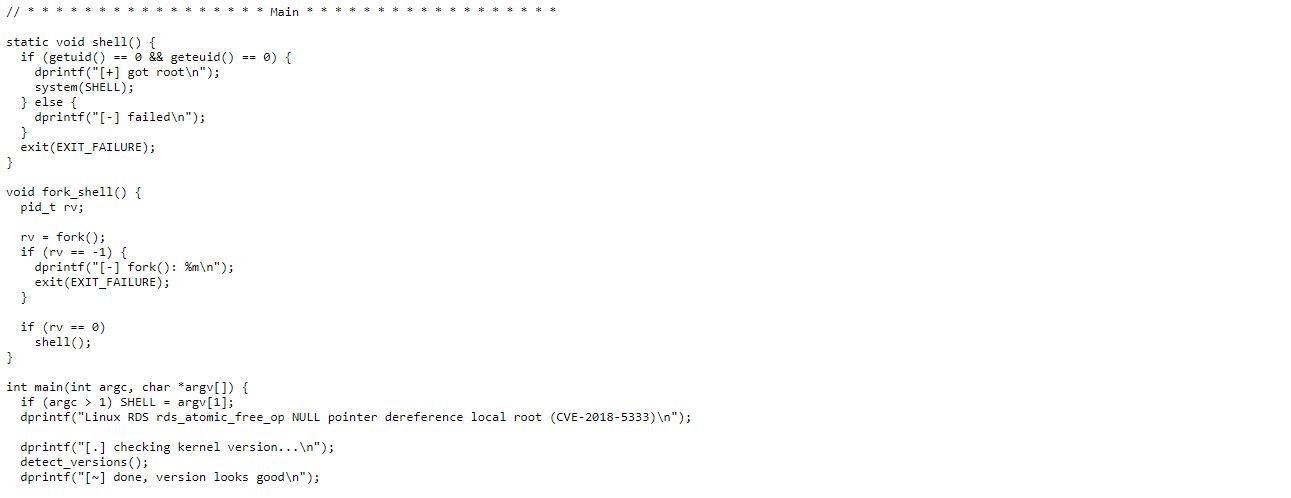


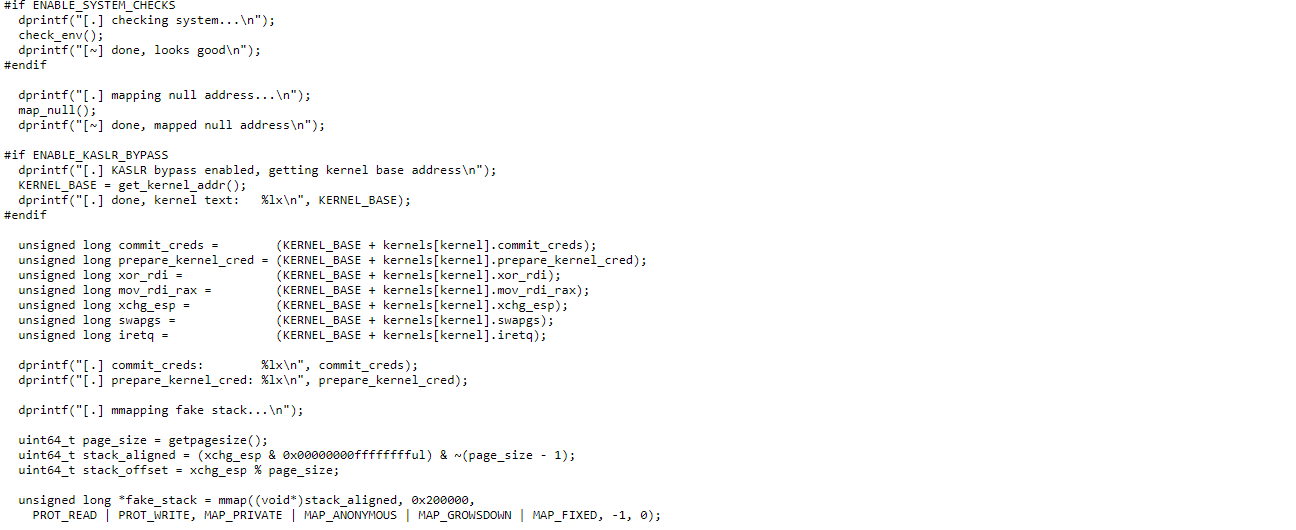


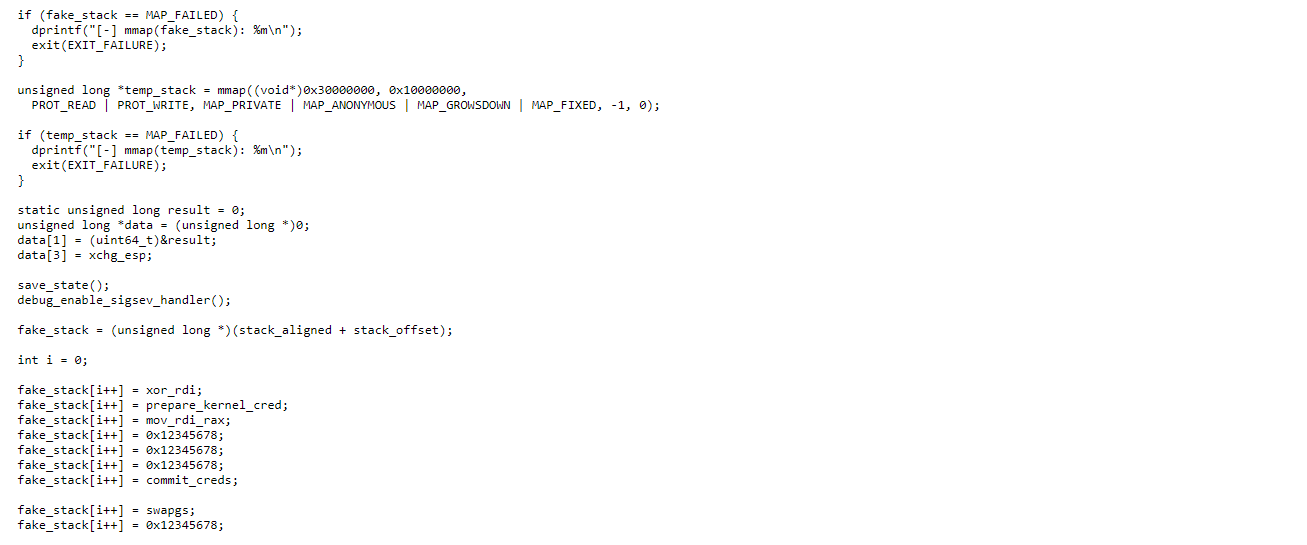














# 4. Methodology

In the methodology,

First, we have to insert the code into the targeted machine, I used ubuntu 16.04.06 with kernel 4.4.5 as the target pc. This kernel is within the vulnerable kernel rage which mentioned above. Hence, I installed it in my virtual box.

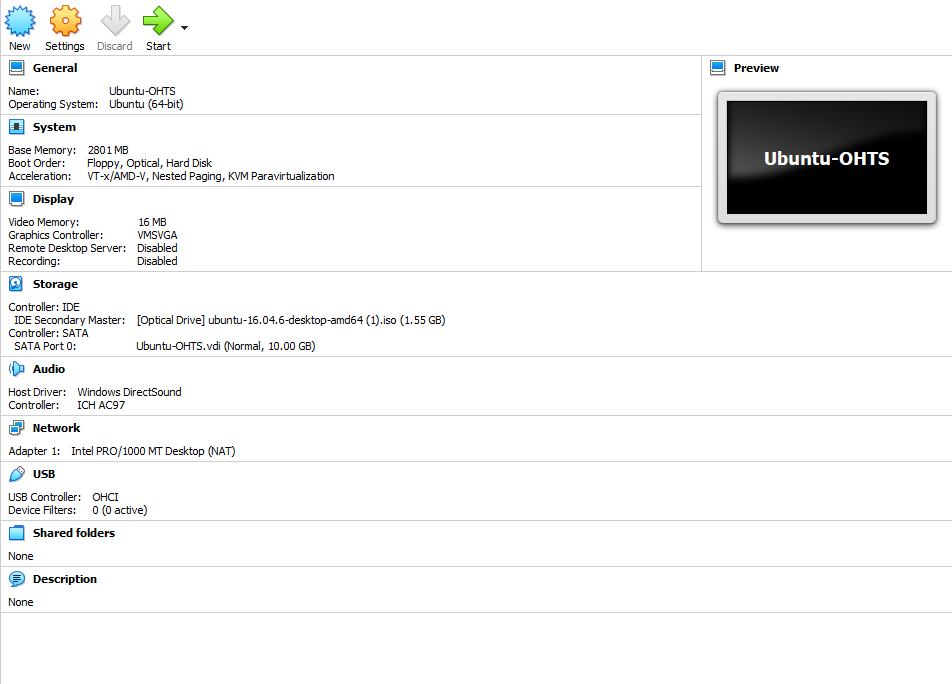


Figure 4.0: targeted virtual machine.

Now, power on the virtual machine.

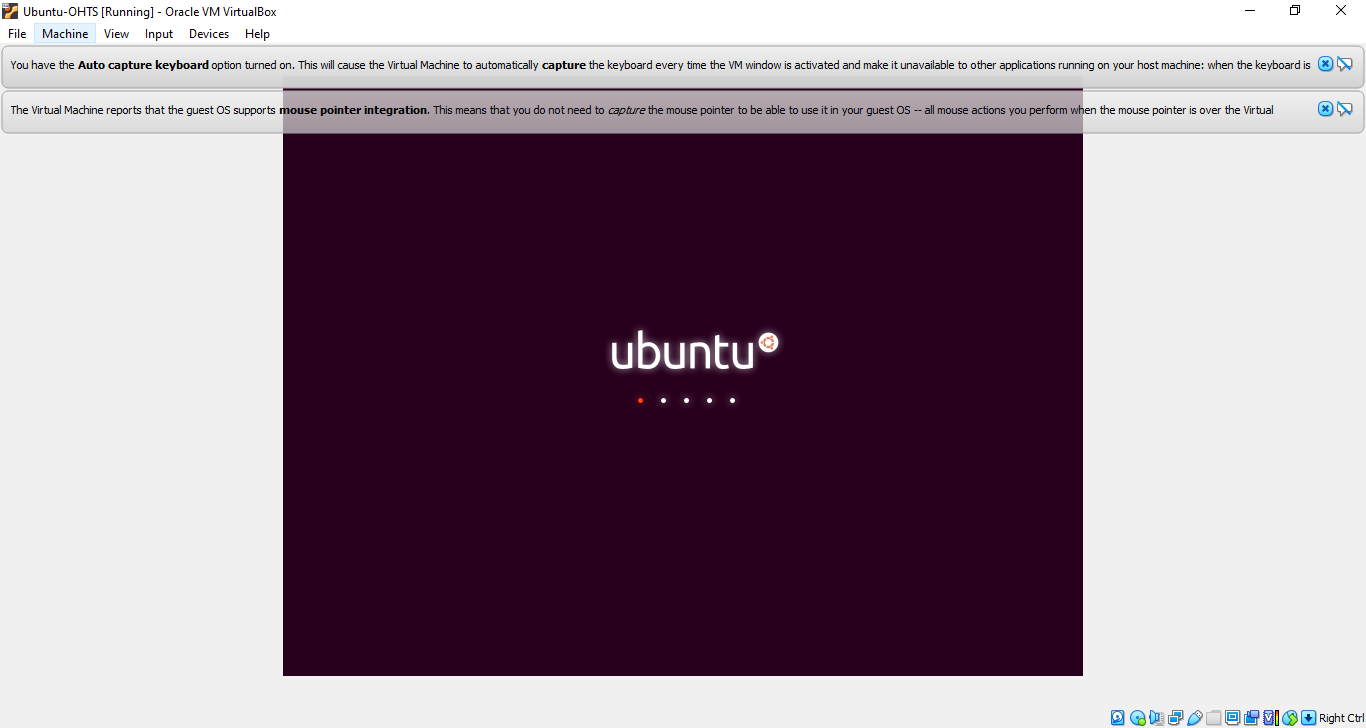


Figure 4.1: Power on the VM.

Then, insert the code to the virtual machine, we can use various kind of methods to do this. I choose to use an USB drive. After I copied and paste it inside the Home directory and opened terminal there.

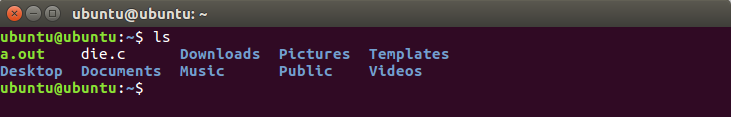


Figure 4.2: Copied exploit.

First install following by typing,

sudo apt install "linux-image-extra-$(uname -r)-generic"

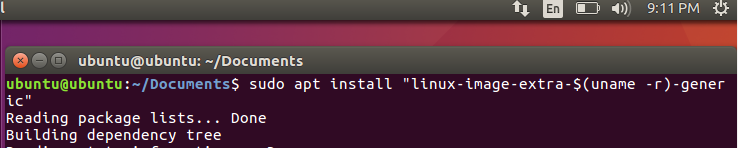


Figure 4.3: Installing requirements.

Then, load following,

sudo insmod "/lib/modules/$(uname -r)/kernel/net/rds/rds.ko"

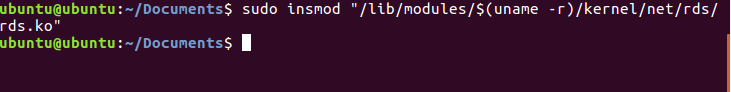


Figure 4.4: Loading requirements.

Now compile the file which we copied as below figure 4.5.

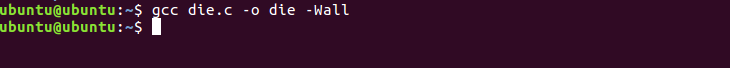


Figure 4.5: Compiling the c file.

Now run the executable c file,



Figure 4.6: running the file.

The outcome.

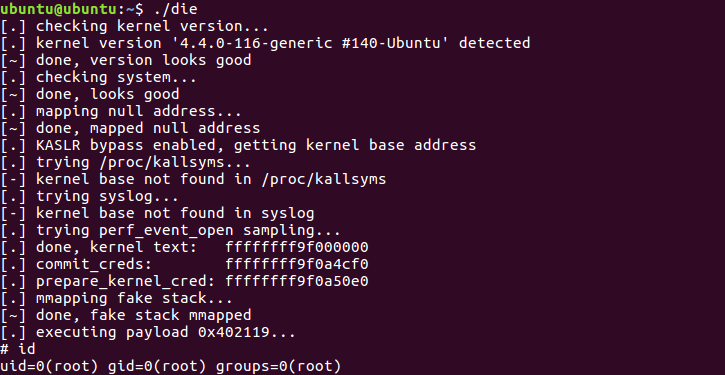


Figure 4.7; Outcome.