# Malicious LKM's and how to detect them

**Patrick Collins**

**CMP408**

## Introduction

Linux Kernel Modules (LKM's) can be created for malicious intentions. Through the widely available cloud solutions such as Amazon Web Services (AWS) nowadays it can be utilised by an attacker to set up an easy attack on the user.

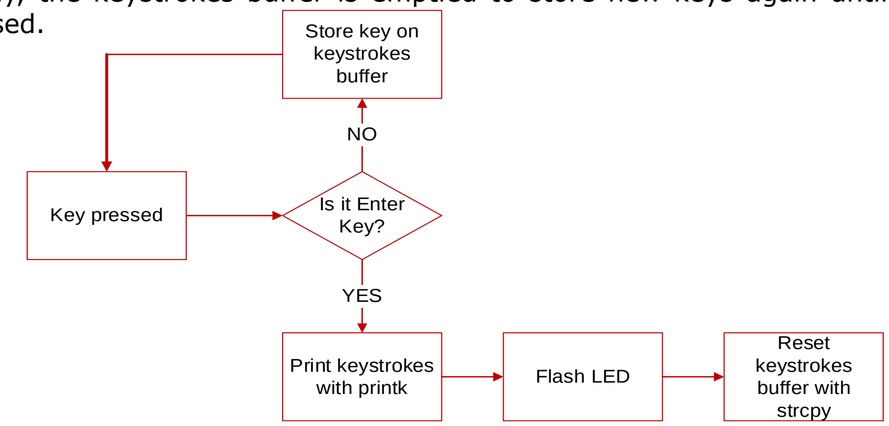
This project is a simple demonstration of how these components could be combined for malicious intent and how to discover such activity.

**Objectives**:

- Set up raspberry pi zero W with Raspbian OS installed
- Create a keylogging LKM module in C
- Turn on LED once post request is sent
- Import this module into the kernel
- Set up Amazon Elastic Beanstalk
- Create PHP website to parse POST data sent to it
- Ensure malicious user input prevention
- Display the user input data on the index.php website page
- Type input into raspberry Pi and refresh the PHP website.
- Download LKM rootkit discovery tool(s) and run on the system to demonstrate how to find malicious LKM

## Methodology

Once a user presses a key the module checks if the key is Enter. If the key is not Enter then it is added onto the string buffer. However, if the user has pressed Enter the keystrokes were intended to be sent to AWS and the LED is flashed to alert the user. However, due to difficulties the keystrokes are instead printed to the kernel. Finally, the keystrokes buffer is emptied to store new keys again until Enter key is pressed.




**LKMKeylogger.ko Flow Chart**

```
static const char* usb_keyboard_scancodes[62] = {
    0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e,
    0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c,
    0x3a, 0x1e, 0x1f, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x2b,
    0x56, 0xac, 0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36,
    0x1d, 0x7d, 0x38, 0x39, 0x64, 0x61, 0x69, 0x67, 0x6c, 0x6a};

static const char* usb_keyboard_shift_scancodes[62] = {
    0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e,
    0x8f, 0x90, 0X91, 0X92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99, 0x9a, 0x9b, 0x1c,
    0x3a, 0x9e, 0x9f, 0xa0, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa7, 0xab,
    0xd6, 0x2c, 0xad, 0xae, 0xaf, 0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6,
    0x9d, 0xfd, 0xb8, 0xb9, 0xe4, 0xe1, 0xe9, 0xe7, 0xec, 0xea};

static const char* convert[62] = {
    "1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "-", "=", " DELETE ",
    " TAB ","q", "w", "e", "r", "t", "y", "u", "i", "o", "p", "[", "]", " ENTER ",
    " CAPS ", "a", "s", "d", "f", "g", "h", "j", "k", "l", ";", "'", "#",
    "\\","z", "x", "c", "v", "b", "n", "m", ",", ".", "/", " SHIFT ",
    " LCtrl ", " PI ", " Alt ", " ", " Alt Gr ", " RCtrl ", " LEFT ", " UP ", " DOWN ", " RIGHT "};

static const char* convertShift[62] = {
    "|", """, "£", "$", "%", "^", "&", "*", "(", ")", "_", "+", " DELETE ",
    " TAB ","Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P", "{", "}", " ENTER ",
    " CAPS ", "A", "S", "D", "F", "G", "H", "J", "K", "L", ":", "@", "~",
    "\\","Z", "X", "C", "V", "O", "N", "M", "<", ">", "?", " SHIFT ",
    " LCtrl ", " PI ", " Alt ", " ", " Alt Gr ", " RCtrl ", " LEFT ", " UP", " DOWN ", " RIGHT "};
```

**UK USB Keyboard Scancodes Mapped**

## Project Highlights

The project was overall successful, combining hardware and software IOT components.

- A malicious LKM Keylogger was created with all user entered keys being stored.
- On Enter press, an LED flashed.
- Rootkit scanning tools didn't find suspicious activity.
- User input could have been sent remotely.

```
[  146.476884] LKMKeylogger: loading out-of-tree m
[  146.484281] Keylogger Loaded
[  153.784744] Keystrokes:[dmesg ENTER ]
[  153.784769] Successful
[  171.576820] Keystrokes:[ CAPS H CAPS ello  CAPS
[  171.576850] Successful
[  257.081053] Keystrokes:[password123 ENTER ]
[  257.081083] Successful
```

**Keystrokes Printed to Kernel**

## Future Work

- Continue implementing functionality to send the keystrokes to a remove webserver.
- Retest rootkit scanning tools with POST requests being sent from Keylogger LKM.

## References

Brouwer, A n.d., *Keyboard scancodes: Keyboard scancodes*, viewed 12 January, 2023, <https://www.win.tue.nl/~aeb/linux/kbd/scancodes-1.html>

Savard, JJG n.d., *Scan Codes Demystified*, viewed 12 January, 2023, <http://www.quadibloc.com/comp/scan.htm>

Dunlap, R and Murray, A n.d., How to get PRINTK format specifiers right¶, *How to get printk format specifiers right - The Linux Kernel documentation*, viewed 13 January, 2023, <https://docs.kernel.org/core-api/printk-formats.html>