Physical access: USB Rubber Ducky Attack (aka. Malduino or BadUSB)

João Loureiro
Security for Networks and Systems
Telecommunications, Eletronics and Computers
Master in Electrical and Computer Engineering
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, 4200-465 PORTO
Email: up201604453@fe.up.pt

José Dias
Security for Networks and Systems
Telecommunications, Eletronics and Computers
Master in Electrical and Computer Engineering
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, 4200-465 PORTO
Email: up201606774@fe.up.pt

***Abstract*—Keystroke injection was first implemented in 2008 and later on perfected in 2010. Imagine plugging in a seemingly innocent USB drive into a computer and installing backdoors, exfiltrating documents, capturing credentials etc.. The sky is the limit from then on, because there is currently no protection from HID Keyboard interfaces. The USB Rubber Ducky disguises itself as system keyboard (keystrokes that can type up to 1000 words per minute!) and has now many names such as Malduino, BadUSB and also many variations when it comes to the build itself, but in most cases it just looks like a USB flash drive, network adapter or even a USB Cable. We investigated further and came to the conclusion, that there aren't really many ways of countering such an attack however surely there is some ways of doing it. We will talk about some the methods we developed as well as some third-party software that we found on the Internet.**

I. Introduction

This work was carried out under the course of Security for Networks and Systems and had the objective of understanding how this type of malicious USB behave, how do they work and how do we protect ourselves from them.

The Rubber Ducky is basically a memory stick (with processing and storage capabilities) that has been modified to show as an HID Keyboard to the target Operative System (HID - Human Interface Device). It started as something you only see in TV shows but now there is open source code and some handy already written scripts all over the Internet, including a Website that compiles Ducky Script into native Arduino-C language. This is because nowadays most Rubber Duckies are programmed in some kind of Arduino model such as Leonardo (or any kind of device with the ATmega32u4 chipset), which comes with HID capabilities out of the box, even Mouse and Joystick).

This is possible to do with any board with minimal memory storage such as the Digispark ATtiny85 Board or even flashing older Arduinos' firmware with a HID Keyboard firmware, this has the inconvenience that you'll need an external library as well, to be able to compile your code into the processor (we chose this path because all we had was a Arduino UNO R3).

II. Related work

As mentioned above, we used a Arduino UNO R3 for the purpose of this assignment. Therefore we needed to flash it with a third-party HEX file that turned the Arduinos' UNO chip (ATmega16u2) into a HID Keyboard injector using the software "Atmel FLIP 3.4.7".

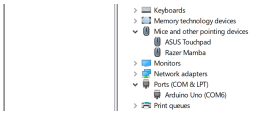

Fig. 1. Serial mode (normal mode of a Arduino UNO).

Fig. 2. DFU - Device Firmware upgrade mode.

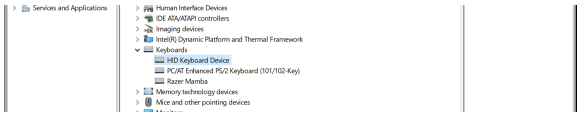

Fig. 3. Arduino finally in HID Keyboard mode.

Only problem with doing this is that you may run into some compatibility problems such as the keyboard nationality layout (Key's scan codes vary from country to country), because the external Arduino libraries are mostly written for US/ENG Keyboards.

So anytime we wanted to test a new script, the UNO R3 had to be flashed back to serial-mode, which is its normal mode, then compile the code into the board for we need to flash it one more time back to HID-keyboard mode. Also, as it has been implied that we had to change our keyboard layout

to US in order to test our scripts (even by doing this, it was not working until the external Arduino library was slightly modified).

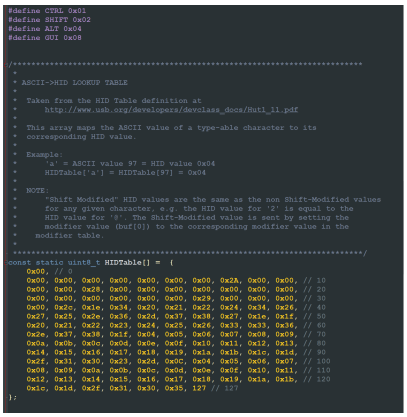

Fig. 4. HIDKeyboard.h lookup table.

On the other hand this wouldn't be necessary in Arduino Leonardo for example as it already has the Arduino's "Keyboard.h" library, and eliminates all of those compatibility problems as it is way more complete than other third-party libraries.

It is important to mention that the focus of this assignment is more about the protection against the device than the variety of attacks that are possible using such a dangerous tool. Although, we thought the concept was so incredible so we tested 2 different scripts:

1) A powershell based script that downloaded a image from the web and opened it;

2) A litlle heavier powershell script that ran a reverse TCP script written in powershell and coded in ASCII-32 (to throw off some of the console detection mechanisms) made in a program called "Veil" and modified by other tools provided by the Kali Linux OS in order to make it undetectable to Windows Defender. Below is the code foe the first example, and as you can see it's a very powerful yet also very simple script.

```
void setup()
{
  keyboard.begin(); //inicia o HIDUINO
  delay(1000);

  keyboard.pressKey(GUI, 'D');
  delay(50);
  keyboard.releaseKey();


  //abrir powershell com admin rights
  keyboard.pressKey(GUI, 'R');
  delay(50);
  keyboard.releaseKey();
  delay(200);
  keyboard.println("powershell Start-Process powershell -Verb runAs"
  );  //Abre a powershell com admin rights
  //keyboard.pressSpecialKey(ENTER);
  delay(50);
  keyboard.releaseKey();
  delay(3000);
  keyboard.pressKey(ALT, 'Y');  //Escolhe "YES" quando o WIndows
  pergunta se tem a certeza que quer aceder com admin rights
  delay(50);
  keyboard.releaseKey();
  delay(2000);
  keyboard.print("$down = New-Object System.Net.WebClient;  $url = '
  https://download941.mediafire.com/gu1u89c00ifg/lzyfdnldluy1s7c/eliot
  .jpg'; $file = 'eliot.jpg'; $down.DownloadFile($url,$file); $exec
  = New-Object -com shell.application; $exec.shellexecute($file);
  exit;");
  keyboard.pressSpecialKey(ENTER);
  delay(50);
  keyboard.releaseKey();
```

Fig. 5. Code for the first case scenario.

The target OS will be Windows 10 version 1909, even though we think it would behave the same for all other versions above 1909. This is because the turning point is just whether the Windows Defender detects the malicious reverse TCP script or not (script nr. 2). As for script nr. 1 there should no problems.

III. Solutions

Being a physical access attack makes it significantly harder to defend from like in any other physical access attack since currently it is impossible for an antivirus to scan a USB firmware in real-time. This is an hot topic in this branch of systems' security and these projects are starting to take off using Machine Learning and AI's which analyzes the input from the USB socket on a very low level.

The best and most important approach is to be very careful with your PC ports and never leave your PC unlocked in public areas with no password whatsoever.

However a more effective solution would be software made to count the speed and flow of the keystrokes and locking your PC in case it senses that it not a human typing. That's exactly what we've done.

A simple Python script was written (redfoxv2.py) in order to take the system to lock-screen, if it detected inhuman keystroke speeds. Off-course the attacker could alter the Rubber Ducky functioning so it would type a lot slower and this software would not work.

The downside for the Python script is that it adds a heavy load on the CPU.

```
from pynput.keyboard import Key, Listener
import logging
import time
import ctypes
#.pyw fica invisivel
prevTime = 0
strikes=0
lastrush=0
var=0

def timeElapsed(start, end):
    delta = end - start
    return delta

# def stop(key):
    # if key == Key.esc:
        # return False

def keypress(Key):
    global prevTime
    global strikes
    global lastrush
    global var
    now=time.time()
    diff=timeElapsed(prevTime, now)

    if prevTime == 0:
        prevTime = now;

    if diff != 0 and diff <= 0.020:
        strikes+=1
        lastrush=time.time()

    prevTime = now;

    if strikes >= 5:
        ctypes.windll.user32.LockWorkStation()
        strikes=0

    var=timeElapsed(lastrush, now)
    if var >= 3:
        strikes=0

with Listener(on_press = keypress, on_release=stop) as listener:
        listener.join()
```

Fig. 6. Code for redfoxV2.py.

Next, we have Penteract Disguised-Keyboard Detector which is a payed application, available on the Microsoft Store.

This covers the keystroke speed countermeasure, as well as USB and Network Adapters white and black listing. So after it detects and locks the screen the first time, it should not work again on the same computer as long as the Rubber Ducky keeps its VID-PID pair and the software is still running (A USB device that is plugged in identifies itself by its VID/PID combination).

The downside for the application would be that by plugging in a HID device (such as a mice) it would still cause the OS to go to lock screen. However, it makes it easier for the user to understand and also tightens up the system's security overall. The computer goes to lock screen for every new device connected, so the user will always know whether it was on purpose or not.

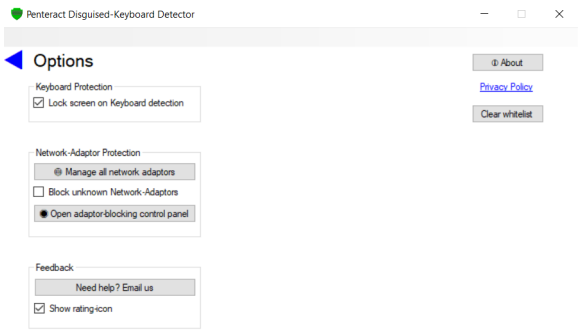

Fig. 7. Penteract's main page.

Finally there some more imaginative ways to stop it, like altering the registry tree in order for the PC to run a captcha every time a USB device is plugged in (we thought of this one too) but this method is extremely expensive if you have a lot of peripheral devices. Also changing values from the registry is never a good idea and you know it if you have ever used Windows. So if this method was to be implemented it should not have to edit anything from there. The final one is the most obvious, which is blocking all command lines, terminal and shells, like cmd and powershell and so on. With these blocked it would be much more difficult for the attacker to get things done, even so because Rubber Duckies are not very dynamic that means if the code doesn't run you have to reprogram it all over again and try a different method.

IV. Evaluation

We managed to stop the threat, but as mentioned before, if the attacker (for some reason) is already familiar with redfoxV2.py or Penteract or any other anti Rubber Ducky techniques it would still be very easy for it to cause damage, but that's the way online security works.

We are confident that at this point a combination of some of these methods should be enough to stop an attack.

We believe we exhausted all our options and discovered that this is still a very new exploit that should not be taken lightly in consideration especially if you run a company or a store with unprotected USB sockets (even if it already exists for more than 10 years). We really enjoyed researching this topic and quickly came up with a few ways to block it but the reality is most people don't know about this, which makes it a very dangerous tool indeed.

V. Conclusion

So there are a couple of ways to protect our PC against Rubber Ducky attacks.

We think the best solution is a mix of them all. Especially the first one, always give your devices a secure password and don't let it sit in public alone.

Also the most effective one would be to disable or protect with password all consoles, shells and terminals (even in admin) so the attacker will have an harder time trying to deceive the OS. The fact that Microsoft is now aware of this didn't help either, for example, (and we tried a lot) now it is impossible to disable Windows Defender with simple keystrokes, you can still navigate but the RETURN (ENTER) does not work as it needs the mouse to change it's value.

Finally we recommend to use either our script (redfoxV2.py) or "Penteract Disguised Keyboard", in combination with the first rule that we talked about (Password!) and locking all consoles with passwords as well. This way is almost impossible for an attacker to break in.