1-1-2023

Bypassing Multiple Security Layers Using Malicious USB Human Interface Device

Mathew Nicho
Rabdan Academy, Abu Dhabi

Ibrahim Sabry
Zayed University

Bypassing Multiple Security Layers Using Malicious USB Human Interface Device

Mathew Nicho[1][a] and Ibrahim Sabry[2][b]
[1]*Research and Innovation Centre, Rabdan Academy, Abu Dhabi, U.A.E.*
[2]*College of Technology Innovation, Zayed University, Dubai, U.A.E.*

Keywords: Arduino, USB, HID, Administrative, Controls, Bypass, Payload.

Abstract: The Universal Serial Bus (USB) enabled devices acts as a trusted tool for data interchange, interface, and storage for the computer systems through Human Interface Devices (HID) namely the keyboard, mouse, headphone, storage media and peripherals that use the USB port. However, with billions of USB enabled devices currently in use today, the attacker's potential to seamlessly leverage this device to perform malicious activities by bypassing security layers presents serious risk to systems administrators. The paper thus presents a comprehensive review of the multiple attacks that can be leveraged using USB devices and the corresponding vulnerabilities including countermeasures. This is followed by the demonstration of five attacks to validate the threat and the associated vulnerabilities by bypassing four security layers namely (1) two server operating system (OS) controls, (2) one group policy control, and (3) antivirus. The attack was performed by plugging in a USB that is connected with the Arduino Micro board to install three differently crafted malwares into the victim machine (Windows Server 2012). As a result, the Arduino device that has been programmed to act like a Human Interaction Device (HID) was able to bypass all the four layers successfully, with execution on the first three layers. The attack-vulnerability theoretical model, the demonstration of the five attacks, and the subsequent analysis of the attacks provide academics with multiple domains (countermeasures) for further research, as well as practitioners to focus on critical IT controls.

1 INTRODUCTION

The threat posed by maliciously modified Universal Serial Bus (USB) devices is real, potent, unsuspecting, relatively easy to execute with deadly consequences, and with a high rate of success. The USB is the most commonly used standard in 5G generation computer systems for peripheral communications (Singh, Biswal, Samanta, Singh, & Lee, 2022). Attackers can modify the firmware of a USB device to masquerade a generic USB flash drive to act as an attacker-controlled, automated, mouse and keyboard (Cronin, Gao, Wang, & Cotton, 2022). In January 2022, the FBI issued a public warning over a USB attack campaign in which numerous USB drives, embedded with malicious codes, were sent to employees at organizations in the transportation, defense, and insurance sectors between August and November 2021 (Hill, 2022). From a global perspective, more than half (54%) of global organizations reported USB-based attacks in 2021, up more than 15 percent from 2020 (Rose, 2022). USB device malware is being leveraged as part of larger cyber-attack campaigns against industrial targets such that 81% of malware seen on USB drives in industrial facilities can disrupt industrial control systems (Honeywell, 2022). From an infection perspective, removable media were responsible for nine percent of all incidents responded to in January 2022, increasing to 20 percent for incidents where the initial infection vector involved a physical endpoint (Mir, Wong, & Manahan, 2022). USB connections and associated devices are inherently trusted (Davis, 2011) and assumed secure by the users partly due to assumed secure ownership and physical proximity of the device (Yang et al., 2015).

The USB enabled peripherals have become an attractive tool for launching cyber-attacks (Nissim, Yahalom, & Elovici, 2017) since a programmable USB device can be used to masquerade as a Human

[a] https://orcid.org/0000-0001-7129-3988
[b] https://orcid.org/0000-0001-9886-9414

Nicho, M. and Sabry, I.
Bypassing Multiple Security Layers Using Malicious USB Human Interface Device.
DOI: 10.5220/0011677100003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 501-508
ISBN: 978-989-758-624-8; ISSN: 2184-4356

Interface Device (HID) (Lawal, Gresty, Gan, & Hewitt, 2021). With billions of USB devices currently in use, the USB protocol is among the most widely adopted protocols today due to its plug-and-play capabilities and the vast number of devices which support the protocol (Denney, Erdin, Babun, Vai, & Uluagac, 2019). As the USB port on a computer system can be considered an open port, it gets exposed to malware with weakened defenses from anti-virus, or security control (Wahanani, Idhom, & Kurniawan, 2020). Furthermore, attackers can design malicious USB devices so that once the USB handshake is completed, and malicious scripts or activities can be executed on the host (Cronin et al., 2022).

The HID that is a component of the user interface system can be exploited using a micro controller embedded with a malicious code to perform malicious actions leading to Malicious HID (MHID). Commonly used HID devices can be identified by many Operating Systems (OS) without the need for specific setup or configuration. Furthermore, the concept of Plug-and-Play (PnP) feature in HIDs (Techopedia, 2019) amplifies the attack surface thereby exposing the vulnerability. Despite the critical nature of USB device attacks, papers focusing on a holistic perspective of analyzing multiple vulnerabilities associated with HID devices is lacking in this domain.

This research takes an exploit and vulnerability approach by presenting a taxonomy of vulnerabilities attributed to HID via the USB protocols through the demonstration of successful attacks bypassing multiple security layers. In this respect, this paper makes the following contributions.

- Provides an overview of attacks on the USB ecosystem by identifying the computer based attacks and vulnerabilities.
- Demonstrates a simulated attack on a server bypassing multiple security layers.
- Analyses the vulnerability associated with the four security layers with suggested countermeasures.
- Demonstrates the criticality of the threat to security practitioners through the exploited vulnerabilities to take appropriate measures.

The paper is structured as follows. Section two and three discusses existing research on HID based attacks and vulnerabilities. Section four and five illustrates the HID based simulated attacks and corresponding countermeasures followed by a final section concluding the research.

2 USB ATTACKS USING HID

Attackers have used USB peripherals to launch cyberattacks, exploiting the vulnerabilities, properties, and capabilities of these devices (Davis, 2011). In this respect, the USB protocol is the most widely-used connector to connect a range of peripheral devices to computers (Neuner, Voyiatzis, Fotopoulos, Mulliner, & Weippl, 2018). It uses a tiered-star topology, where the center of the star is the USB host, which defines the USB, that is implemented as a combination of hardware, firmware, and software residing inside the computer.

The HIDs have elevated privileges, because the systems' OS assumes that commands executed from the HID are coming from an authorized user. Altering a USB device to simulate keyboard or mouse is a method to emulate an HID by hiding malicious codes for malicious actions (Nasution, Purwanto, Virgono, & Alam, 2014). Attack techniques using peripheral devices may use USB devices by emulating a USB Ethernet adapter. This enables the USB device to act as a DHCP server that directs traffic through a malicious DNS (Tischer et al., 2016). Attacks deploying social engineering methods leveraging USB are aimed at organizations to make the attack happen (Anderson & Anderson, 2010; Davis, 2011; Pham, Syed, & Halgamuge, 2011).

The USB Rubber Ducky is a keyboard emulator hidden within a USB thumb drive case. Since 2010, IT professionals, penetration testers, and hackers have used the Rubber Ducky to create the most widely used commercial keystroke injection attack (Nissim et al., 2017). BadUSB attacks are initiated when the host performs a malicious firmware update in which the USB device's firmware is modified by the attacker where it acts according to how it was programmed by the attacker (Cannoles & Ghafarian, 2017). Another type of USB-based attack is a driver-related attack, in which an attacker inserts a compromised malicious USB device into the host, causing the host to download a malicious driver crafted to execute malicious code or exploit a buffer overflow vulnerability (Caudill & Wilson, 2014). The attack methodology deployed for the use of bad USBs is through reprogramming the device's USB protocol stack. Using micro controller devices namely the Raspberry pi Pico, Intel 8051, the micro controller's logic circuit programming is altered, and the primal functioning of a simple IoT device is transitioned to be used in cyber-attacks. There are also attacks that are oriented toward the Denial of Service (DoS) of the host computer. Once such attack is termed the USB Killer attack where a look alike flash drive acts as an

electric discharger capable of destroying sensitive components on the host (Nissim et al., 2017). This attack was successfully demonstrated through the 'Stuxnet' worm attack.

3 USB BASED VULNERABILITIES

Attacks exploiting USB ecosystem vulnerabilities are varied (see figure 1) with the most recent and potent being the Rubber Ducky. The Rubber Ducky vulnerability (Nohl & Lell, 2014) that is demonstrated in this research makes use of the weak threat detection by the OS and host machine, even with various controls in place. This is possible due to the configuration flexibility provided by the Arduino microcontroller in order to run malicious scripts that can control the host machine through the input of PowerShell commands embedded into the Arduino C++ script, resulting in the download and execution of the malicious payload. Since, USB devices are innocuous by their nature and ubiquity, their architecture of the USB ecosystem is 'complex' and deeply 'embedded' in the OS. Furthermore, the features of the USB protocol that resemble wide-area networking protocols can be 'underappreciated'. This combination of complexity, embeddedness, and under appreciation is the foremost vulnerability feature of MHID (Johnson, 2014).

3.1 Encryption & Data Misplacement

Attacks by MHIDs exploiting the OS in computer systems is a serious risk due to the inherent trust given by the OS to HIDs. In this respect, the use of malicious HID devices has exposed critical vulnerabilities in OS (Zhao & Wang, 2019). HID vulnerabilities that can be exploited include poor encryption and data misplacement, systems misconfiguration, unpatched and outdated software. Since HIDs are not inherently malicious, USB device firmware (drivers) are not typically scanned by OS. In addition, OS embedded antimalware systems are unable to detect or defend against these types of attacks. Therefore, HIDs are more susceptible to manipulation as they gain seamless access to the host machine. Once connected to the computer system, they can pose as a peripheral such a mouse or keyboard or download the malicious payload or execute a code on the host machine that would further weaken the system. The USB protocol vulnerability that could allow a USB device to impersonate peripheral devices occurs due to the lack of device authentication, authorization and a security mechanism that needs to verify the data or the messages received from a device (Ticu, 2021). A Man in the Middle (MITM) attack can also be effected using MHID due to the lack of appropriate end-to-end encryption. Hence, when a MHID is plugged in (Ex. Rubber Ducky), it can tap into the duplex communication pathway and steal sensitive information (Acar, Lu, Uluagac, & Kirda, 2019).

3.2 Systems Misconfiguration

This vulnerability is portrayed by system assets such as operating systems where their internal settings may be vulnerable. In addition, the application or operating system settings may be disparate in terms of security (Scaife, Peeters, & Traynor, 2018). Additionally, while running the research experiment it evident that the policy integrated into the GPO is to deny access of "all removable storage classes", which means that it does not include removable peripheral HIDs namely keyboard and mouse.

3.3 Unpatched/Outdated Software

Outdated OS pose serious threat in relation to the use of malicious HID devices since the unpatched software is outdated with less security update support from the vendor. Thus, cybercriminals will tend to track users that use unpatched or outdated system software as it will be easy to compromise and to access and retrieve sensitive information and data (Zhao & Wang, 2019). Figure 1 summarizes the vulnerabilities and the associated attack vectors.

4 SIMULATION AND ANALYSIS

In this simulated experiment, we target the USB protocol function drivers namely the device function, function driver, and device controller driver. We opted to go with the Arduino Micro board, as it is one of the boards with built in USB capabilities, that would allow us to use its connection to act as an external keyboard or mouse when connecting to a machine. Thereafter, we used a male-to-male USB adapter which allows for easy plug and play functionality to the Arduino board. To upload the script onto the board we used the Arduino IDE environment that acts as a communication bridge between the host machine and the board.

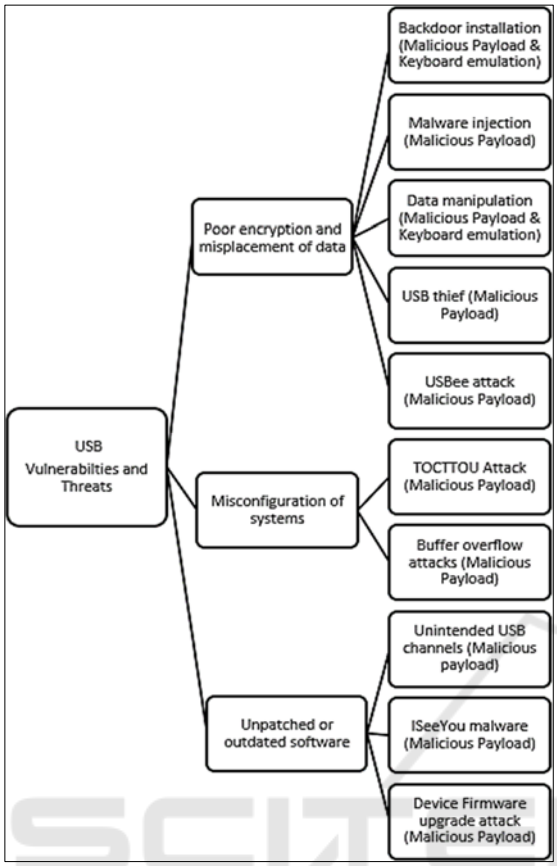

Figure 1: USB Vulnerabilities exploited by HIDs.

4.1 USB Rubber Ducky Set up

In this paper, we selected the Rubber Ducky option rather than the BadUSB, as the BadUSB can only be executed on USB 3.0 flash drives that contain the Phison 2303 micro-controller (with limited availability) while the Rubber Ducky is a lot faster than BadUSB and a Teensy and is readily available. It does not contain any embedded malware, but only executes commands to download it from specific IP address via the victim's browser. The environment set up for the experiment consists of two virtual machines, namely the Kali Linux VM (attacker), and Windows Server 2012 (victim). The Server version of Windows was used in order to create an active directory, and simulate a direct attack on the Server to gain administrative controls while logged in as the admin (Figure 2).

In order to masquerade Arduino as an HID, we connected the Arduino and configure it using the IDE interface. The IDE interface allows the user to create scripts using C++ programming language, which was uploaded into the Arduino board to execute upon reconnecting the board to the machine. This converts the Arduino into a "rubber ducky" to take control of the keyboard and uses it to run commands on the victim machine.

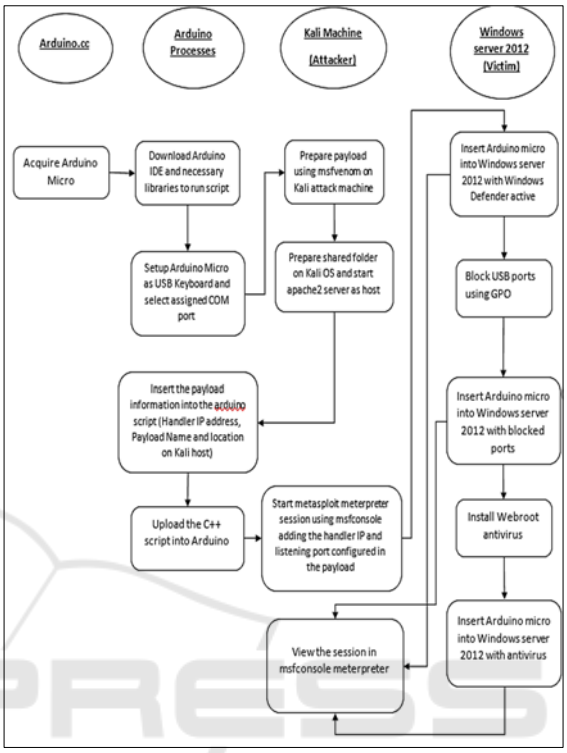

Figure 2: Attack flowchart.

In the C++ script we also inserted a script to disable the Windows Defender (Box 1).

Box 1: Script to disable Windows Defender.

```
Keyboard.print("powershell start
powershell -A 'Set-MpPreference -
DisableRea $true' -V
runAs");  typeKey(KEY_RETURN);
```

We also inserted the script to download the malicious file via the victim's browser (open the web client and download the file from the Kali Apache server) and execute it (Box 2). We ensured that the script libraries for the keyboard are available, so that the Arduino would be able to connect as a Rubber Ducky. This can be seen with the *#include 'Keyboard.h' function*, which allows the Arduino to gain access to the keyboard library scripts. The *Keyboard.press* and *Keyboard.release* functions control the victim machine's keyboard in order to type in the intended commands on PowerShell. Thereafter, we created the malicious Windows payload to open a reverse_tcp connection using msfvenom.

Box 2: Script for accessing and downloading the malware from an external IP address.

```
Keyboard.print("powershell -
windowstyle hidden (New-Object
System.Net.WebClient).DownloadFile('h
ttp://192.168.100.6//share/whysoserio
us.exe','%TEMP%\\whysoserious.exe');
Start-Process
\"%TEMP%\\whysoserious.exe");
  typeKey(KEY_RETURN);
  // Ending stream
  Keyboard.end();
}
/* Unused endless loop */
void loop()
```

Thereafter, we configured Windows Server 2012 to create the Active Directory in order to simulate the server environment. We ensured that the Windows Firewall and Windows Defender are up and running as in a real environment. In order to simulate the varied security layers in an organizational network we conducted three simulated attacks (A, B, C) on Windows Server 2012. In all first option the attack was done with the two security layers namely Windows firewall, and Windows Defender activated. For B, we added a third security layer by configuring Windows Group Policy Object (GPO) to disable USB ports in the Server. In the last option a fourth security layer was added by installing and activating an antivirus.

4.2 Attack Option A

This option simulates the security structure of Windows Server in a normal network scenario of organizations where the use of USB is allowed. We begin the experiment by plugging in the Arduino board through the USB port and waiting for the script to run. Thereafter, The Arduino automatically downloaded and installed the malware onto the Windows Server 2012 through the URL. This was done without any human intervention via the PowerShell command that was embedded in the Arduino (See box 2). We were able to successfully navigate the system33 folder which would only be allowed for administrator access on the windows machine. We also successfully ran the *clearrev*, *getuid*, *ps* (process list) and *ifconfig* (Network information) commands to simulate malicious actions performed by hackers and get additional information on the machine. The Arduino device was able to bypass two security layers of admin controls in order to download the payload onto the machine. Furthermore, the software was downloaded onto the *%temp%* folder of the windows machine, which was appropriate for the attacker for stealth reasons. This demonstrates the significance of the USB vulnerability, as the Arduino was easily able to bypass admin controls just by connecting to the victim machine as an external keyboard.

4.3 Attack Option B

In this phase of the attack we added a third security layer by configuring the GPO to block USB ports on the server by launching the GPO tool on the domain controller. Once the *gpupdate /force* command to enforce the GPO was executed, we tested the GPO by connecting a USB drive to the OS where the USB was denied (figure 6) access thus validating the enablement of the GPO. When the experiment was restarted by plugging in the Arduino board through the USB port, the script was successful in executing the malware. It's important to note that even though GPO was active, the Arduino micro masquerading as a HID managed to bypass the GPO security control. Subsequently, we repeated the commands in first attack by successfully navigating to the system33, ran the *clearrev*, *getuid, ps* (process list) and *ifconfig* commands, getting the same outputs as in the previous attack. This demonstrate that despite the activation of the GPO blocking USB ports, the Arduino script was successful in taking over the victims' machines' keyboard and execute the programmed powershell commands to download the malicious software from the network and even execute it.

4.4 Attack Option C

In this option, we build on the previous result by adding a fourth security layer by downloading and installing Webroot SecureAnywhere antivirus (listed among the top ten in 2022) onto the Windows Server 2012 VM. When we ran our experiment, the Arduino script was successful in penetrating the four defensive layers. However, the antivirus (fourth defensive layer) detected and blocked the malware when the Arduino script attempted to download and execute the malware, thus preventing the execution of the malware.

This signifies that while the Arduino script was successful in its initial operation, the infection was unsuccessful. In this respect, the antivirus was unable to detect the Rubber Ducky, but detected the malware signature. Since we used msfvenom to create the malware (C1), we repeated the same attack by creating a second malware with FatRat *(Option 6:*

Create FUD Backdoor 1000% with PwnWinds). We simulated the same experiment but with the same result as in C1. Furthermore, we created a third malware with Veil Evasion *(Option 22: poweshell/meterpreter/rev_tcp.py)* and did the same attack with the same result as in C1 and C2. Table 1 thus demonstrates the result of the five attacks. Since, malwares are continuously evolving, so too are antivirus endpoint security solutions. This is a characteristic that makes every malware unique in its signature or fingerprint. This identification information is regularly updated onto the antivirus database and stored as virus definitions This process is done in order to be up to date with new versions of malware signatures/fingerprints. If a match is detected by the antivirus, it alerts the user while also containing the virus in a sandbox environment (Vigderman & Turner, 2022). However, when we attempted to bypass this feature, with a modified script, we were unsuccessful in installing and executing the malware.

Table 1: The five attack configurations with the results.

	GPO	Firewall	W Defender	AV	**Result**
A	x	Yes	Yes*	x	**S**
B	Yes	Yes	Yes*	x	**S**
C1	Yes	Yes	Yes*	Yes	**U****
C2	Yes	Yes	Yes*	Yes	**U****
C3	Yes	Yes	Yes*	Yes	**U****

('Yes' means that the security layer was active; 'x' signifies it is inactive; S denotes that the attack is successful, while U denotes its unsuccessful in executing the malware)

**The Windows Defender was running, but was disabled by the Arduino with the embedded PowerShell script:*
`"powershell start powershell -A 'Set-MpPreference -DisableRea $true' -V runAs"`

*** The Arduino script ran successfully, but the malware was detected as a Trojan.*

In order to counter the fourth defensive layer (bypass and/or disable the Webroot Realtime antivirus), we added a powershell command to the script, which would uninstall the antivirus software. But, this method was unsuccessful due to the software requiring the user to enter an auto generated CAPTCHA code, which is an effective countermeasure to avoid tampering with the antivirus software. However, two methods can bypass the above preventive method. First, a successful penetration can be done using a zero day or near zero-day malware to evade antivirus software. Secondly, through the use of different malware authoring tools to update the malware in the Apache Server folder using the same malware name as in Box 2 (*whysoserious.exe').* With multiple malware authoring tools readily available and with multiple options in each of these tools, evasion is not difficult.

5 COUNTERMEASURES

While, tools and techniques for detecting USB-related attacks have been proposed and implemented, currently there are no fool proof technique for fully guarding against potential USB malicious attacks, as most solutions may be circumvented. As with most security systems, applying specific IT control corresponding to each security layer offers the best chance to increase security as discussed below. The sub sections below provide a categorised discussion on the security that can be applied at the different USB system layers.

5.1 USB Functional Layer

The USBFILTER system, which is a packet-level access control (firewall) for USB, can allow or deny USB device functionality by defining specific measures (Tian, Bates, Butler, & Rangaswami, 2016). The researchers instrumented the host's USB stack between the USB device driver and the USB controller, allowing them to filter packets at the operating system's low level (kernel). Furthermore, USBFILTER can specify which programs (e.g., Zoom) are permitted to use which USB devices (e.g., webcams, speakers, and microphones), preventing malicious software on the host from enabling or accessing protected USB devices (Neuner, 2017). Cinch is based on the same principles as USBFILTER. However, it isolates all USB devices from the host and routes communication through a virtual machine that acts as a gateway, enforcing access policies (Angel et al., 2016). There are certain scripts like DuckHunter that are designed to run in the background and monitor the typing speed. The program effectively blocks keyboard input when a Rubber Ducky attack is detected (Pmsosa, 2020). While a tool such as SandUSB acts as a middleman between the device and connected host, the tool places the connected USB into a sandbox environment to detect any malicious activity, by scanning and analyzing the device. The tool looks at three different aspects that may be detected as malicious such as keyboard typing speed or pattern, any attempts to change OS configuration, and finally the detection of malicious payload on the storage device (Loe, Hsiao, Kim, Lee, & Cheng, 2016).

5.2 USB Physical Layer

USB port blockers are an efficient way to prevent users from connecting unauthorized USB devices with malicious payloads. An attacker is less likely to target systems with USB port blockers in the case of a Rubber Ducky attack. USB port blockers, which come with a special key that unlocks and locks the device once installed, can be installed on critical systems in a network that contains sensitive files. The disadvantage of USB port blockers is that they are still vulnerable to physical tampering if not monitored. Rubber Ducky devices operate at speeds that are impossible to detect by standard users in a company. Various payload scripts are readily available for download from the internet, allowing even those with no computer experience to carry out a Rubber Ducky attack (Karystinos, Andreatos, & Douligeris, 2019).

5.3 USB Link Layer

Commercial Rubber Ducky attack countermeasures makes it impossible for Rubber Ducky devices to emulate a keyboard. When a USB device connects to the computer and the Operating System recognizes it as a keyboard, the anti-virus prompts the user to enter a numerical challenge code generated by the anti-virus from the 'new' USB keyboard. Keyboard authorization is a term used to describe such a procedure. As a result, the anti-virus will only allow the use of a keyboard that has been authorized and will block any other keyboard that has not been authorized. Even if it could not detect the Rubber Ducky device, it can detect the malware launched by the Rubber Ducky (Raghavan, 2020).

From an encryption perspective, commercial solutions exist where if the USB device manufacturer is trusted and the signing key is kept secure, the firmware is considered trusted. Another potential way to prevent the Rubber Ducky attack is by restricting access to elevated Command Prompt (cmd). Running cmd as an administrator unlocks a whole set of actions that can be performed on a computer. That's why an administrator should set a password for using elevated cmd to stop any Rubber Ducky programmed to seek out administrative privileges.

6 CONCLUSION

This paper demonstrated a simulated attack by an USB drive masquerading as a malicious human interface device using Arduino micro-controller. Our device was able to bypass four security layers namely the OS Firewall, the OS Antivirus, the OS security control (GPO), and partly evade the installed antivirus. The identified and analyzed vulnerabilities provide valuable insights into probable countermeasures that can be deployed providing greater visibility of the threat and corresponding countermeasures. This shows the critical nature of USB based HID attacks that can penetrate networks through a malicious or an unsuspecting insider.. The experiment shows the severity of the USB vulnerability.

While our paper is not without its limitations, each of these listed limitation provides avenues for further research. First, since we did a direct attack on a server, an attack simulating a system in a network via a secure remote connection can provide network based vulnerabilities as well. Second, we created malware using three malware authoring tools. With numerous malware authoring tools available to hackers, the fourth security layer can be fully bypassed using zero day or near zero-day malware. Third, we used a script to de activate the Windows Defender. Research on innovative script can deactivate similar antivirus programs. Fourth, the research can be extended to include experimental methods for gaining domain admin users rights in LDAP or active directory and illustrate more about the extent to which admin domain privileged can be controlled. Finally, we did not specifically test the countermeasures against specific attacks for validating the countermeasures.

REFERENCES

Acar, A., Lu, L., Uluagac, A. S., & Kirda, E. (2019). An analysis of malware trends in enterprise networks. Paper presented at the International Conference on Information Security.

Anderson, B., & Anderson, B. (2010). Seven deadliest USB attacks: Syngress.

Angel, S., Wahby, R. S., Howald, M., Leners, J. B., Spilo, M., Sun, Z., . . . Walfish, M. (2016). Defending against malicious peripherals with Cinch. Paper presented at the 25th USENIX Security Symposium (USENIX Security 16).

Cannoles, B., & Ghafarian, A. (2017). Hacking experiment by using usb rubber ducky scripting. Journal of Systemics, 15(2), 6671.

Caudill, A., & Wilson, B. (2014). Making BadUSB work for you. Paper presented at the presented at Derbycon 4.0. Available online from.

Cronin, P., Gao, X., Wang, H., & Cotton, C. (2022). Time-print: Authenticating USB flash drives with novel

timing fingerprints. Paper presented at the 2022 IEEE Symposium on Security and Privacy (SP).
Davis, A. (2011). USB-undermining security barriers. Black Hat Briefings.
Denney, K., Erdin, E., Babun, L., Vai, M., & Uluagac, S. (2019). Usb-watch: a dynamic hardware-assisted usb threat detection framework. Paper presented at the International Conference on Security and Privacy in Communication Systems.
Hill, M. (2022). BadUSB explained: How rogue USBs threaten your organization. CSO Online. Retrieved from https://www.csoonline.com/article/3647173/badusb-explained-how-rogue-usbs-threaten-your-organization.html
Honeywell. (2022). Honeywell Industrial USB Threat Report. Retrieved from Houston: https://honeywellprocess.blob.core.windows.net/public/Marketing/Honeywell-USB-Threat-Report.pdf
Johnson, P. C. (2014). How USB Does (and Doesn't) Work: A Security Perspective. Security, 39(4), 12-15.
Karystinos, E., Andreatos, A., & Douligeris, C. (2019). Spyduino: Arduino as a HID exploiting the BadUSB vulnerability. Paper presented at the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS).
Lawal, D., Gresty, D. W., Gan, D., & Hewitt, L. (2021). Have You Been Framed and Can You Prove It? Paper presented at the 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO).
Loe, E. L., Hsiao, H.-C., Kim, T. H.-J., Lee, S.-C., & Cheng, S.-M. (2016). SandUSB: An installation-free sandbox for USB peripherals. Paper presented at the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT).
Mir, H., Wong, S., & Manahan, B. (2022). Top Attack Vectors: January 2022. Retrieved from https://expel.com/blog/top-attack-vectors-january-2022/
Nasution, S. M., Purwanto, Y., Virgono, A., & Alam, G. C. (2014). Integration of kleptoware as keyboard keylogger for input recorder using teensy USB development board. Paper presented at the Telecommunication Systems Services and Applications (TSSA), 2014 8th International Conference on.
Neuner, S. (2017). Bad things happen through USB. Wien,
Neuner, S., Voyiatzis, A. G., Fotopoulos, S., Mulliner, C., & Weippl, E. R. (2018). Usblock: Blocking usb-based keypress injection attacks. Paper presented at the IFIP Annual Conference on Data and Applications Security and Privacy.
Nissim, N., Yahalom, R., & Elovici, Y. (2017). USB-based attacks. Computers & Security, 70, 675-688.
Nohl, K., & Lell, J. (2014). BadUSB-On accessories that turn evil. Black Hat USA, 1(9), 1-22.
Pham, D. V., Syed, A., & Halgamuge, M. N. (2011). Universal serial bus based software attacks and protection solutions. Digital Investigation, 7(3-4), 172-184.
Pmsosa. (2020). Duckhunt: Prevent Rubberducky (or other keystroke injection) attacks, DuckHunter. Retrieved from https://github.com/pmsosa/duckhunt
Raghavan, R. (2020). An Introduction to Bad USB Attacks. Retrieved from https://acodez.in/badusb-attack/
Rose, A. (2022). Cybercriminals bring the USB back, with a vengeance. Retrieved from https://www.itp.net/opinion/cybercriminals-bring-the-usb-back-with-a-vengeance#:~:text=More%20than%20half%20(54%25),environment%20as%20an%20acceptable%20option.
Scaife, N., Peeters, C., & Traynor, P. (2018). Fear the reaper: Characterization and fast detection of card skimmers. Paper presented at the 27th USENIX Security Symposium (USENIX Security 18).
Singh, D., Biswal, A. K., Samanta, D., Singh, D., & Lee, H.-N. (2022). Juice Jacking: Security Issues and Improvements in USB Technology. Sustainability, 14(2), 939.
Techopedia. (2019, November). What Does Human Interface Device (HID) Mean. Retrieved from https://www.techopedia.com/definition/19781/human-interface-device-hid
Tian, D., Bates, A., Butler, K. R., & Rangaswami, R. (2016). Provusb: Block-level provenance-based data protection for usb storage devices. Paper presented at the Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.
Ticu, M. (2021). USB Traffic Analyzer-digUSB. Paper presented at the 2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE).
Tischer, M., Durumeric, Z., Foster, S., Duan, S., Mori, A., Bursztein, E., & Bailey, M. (2016). Users really do plug in USB drives they find. Paper presented at the 2016 IEEE Symposium on Security and Privacy (SP).
Vigderman, A., & Turner, G. (2022). How Does Antivirus Software Work? Retrieved from https://www.security.org/antivirus/how-does-antivirus-work/
Wahanani, H., Idhom, M., & Kurniawan, D. (2020). Exploit remote attack test in operating system using arduino micro. Paper presented at the Journal of Physics: Conference Series.
Yang, B., Qin, Y., Zhang, Y., Wang, W., & Feng, D. (2015). TMSUI: A trust management scheme of USB storage devices for industrial control systems. Paper presented at the International Conference on Information and Communications Security.
Zhao, S., & Wang, X. A. (2019). A Survey of Malicious HID Devices. Paper presented at the International Conference on Broadband and Wireless Computing, Communication and Applications.