T.R.
USKUDAR UNIVERSITY INSTITUTE OF SCIENCE

DEPARTMENT OF
CYBERSECURITY PROGRAM

MASTER THESIS

CONVERTING NORMAL USB TO BAD USB

Mohammad Sameer ABUSARA

Thesis Advisor
Prof. Dr. Türker ERGÜZEL

ISTANBUL-2024

T.R.
USKUDAR UNIVERSITY INSTITUTE OF SCIENCE

DEPARTMENT OF
CYBERSECURITY PROGRAM

MASTER THESIS

CONVERTING NORMAL USB TO BAD USB

Mohammad Sameer ABUSARA

Thesis Advisor
Prof. Dr. Türker ERGÜZEL

ISTANBUL-2024

Text of the Oath

I declare that this study is my own thesis study, that I have no unethical behavior at any stage from planning to writing, that I have obtained all the information in the thesis within academic and ethical rules, and that I resource all the information and comments that are not obtained through the thesis study.

Date

Name Surname

Signature

SUMMARY

The flexibility that comes along with USB technology has greatly enhanced the storage of data and transmission of data but at the same time creating serious security risks. Malware hiding in USB peripherals is called “Bad USB”, it manages to execute unauthorized commands and infiltrate secure settings since people trust USB peripherals. This thesis focuses on two primary mechanisms of such attacks: Ducky Script and VBScript are also similar to each other and are programming languages designed for automating for Windows operating systems. USB Rubber Ducky uses Ducky Script to configure keystrokes to be injected, meaning that it can allow an attacker to easily input pre-set commands and compromise a target computer. While VBScript can use USB autorun properties to download and run a malicious code, this tends to go unnoticed by the user as the USB is connected. This analysis outlines the basic USB operation for typical and legal purpose such as file sharing and connectivity of other devices with the computer and the operation of Bad USB devices. It also covers how one can actually turn a normal USB drive into a Bad USB, ranging from firmware loading, scripting, testing, and activating. Thus, by drawing attention, such major security issues, the thesis underlines the importance of designing proper detection, protection, and countermeasures against the modern USB-related threats.

Keywords

USB security, Bad USB, Ducky Scripts, VBScript, Cybersecurity, Malware, Exploitation

ÖZET

USB teknolojisiyle birlikte gelen esneklik, verilerin depolanmasını ve aktarımını büyük ölçüde artırdı ancak aynı zamanda ciddi güvenlik riskleri de yarattı. USB çevre birimlerinde saklanan kötü amaçlı yazılımlara "Kötü USB" denir, insanlar USB çevre birimlerine güvendiğinden, yetkisiz komutları yürütmeyi ve güvenli ayarlara sızmayı başarır. Bu tez, bu tür saldırıların iki temel mekanizmasına odaklanmaktadır: Ducky Script ve VBScript de birbirine benzer ve Windows işletim sistemleri için otomasyon amacıyla tasarlanmış programlama dilleridir. USB Rubber Ducky, tuş vuruşlarını enjekte edilecek şekilde yapılandırmak için Ducky Komut Dosyasını kullanıyor; bu, bir saldırganın önceden ayarlanmış komutları kolayca girmesine ve hedef bilgisayarın güvenliğini aşmasına olanak tanıyabileceği anlamına geliyor. VBScript, kötü amaçlı bir kodu indirmek ve çalıştırmak için USB otomatik çalıştırma özelliklerini kullanabilse de, USB bağlandığında bu durum kullanıcı tarafından fark edilmeyebilir. Bu analiz, dosya paylaşımı ve diğer cihazların bilgisayarla bağlantısı ve Kötü USB cihazlarının çalışması gibi tipik ve yasal amaçlara yönelik temel USB işlemlerini özetlemektedir. Aynı zamanda normal bir USB sürücüsünün, ürün yazılımı yükleme, komut dosyası oluşturma, test etme ve etkinleştirme gibi işlemlerle nasıl Kötü USB'ye dönüştürülebileceğini de kapsar. Dolayısıyla tez, bu tür önemli güvenlik sorunlarına dikkat çekerek, modern USB ile ilgili tehditlere karşı uygun tespit, koruma ve karşı önlemlerin tasarlanmasının öneminin altını çizmektedir.

Anahtar Kelimeler

USB güvenliği, Kötü USB, Ducky Komut Dosyaları, VBScript, Siber Güvenlik, Kötü Amaçlı Yazılım, Suistimal

Table of Contents

Text of the Oath i
SUMMARY ii
SUMMARY iii
Table of Contents iv
List of tables vi
List of Figures vii

1. Introduction 1

2. Literature Review 3
2.1 Overview of USB Technology 3
2.1.1 Historical Development 3
2.1.2 Functional Components 3
2.2 USB Security Threats 5
2.3 Bad USB, Ducky Scripts and VBScript 6
2.4 Existing Mitigation Techniques 7

3. Methodology 8
3.1 Normal USB Functionality 8
3.2 Bad USB Ducky Scripts 8
3.3 Bad USB VBScript 8
3.4 Mechanisms of Bad USB Ducky Scripts 10
3.5 Mechanisms of Bad USB VBScript 11

4. Comparative Analysis of Normal USB and Bad USB..... 12
4.1 Comparison between Normal USB and Bad USB 12
4.2 Comparison between Normal USB and Bad USB Ducky Script 13

4.3 Comparison between Normal USB and Bad USB VBScript 14
4.4 Comparison between Bad USB Ducky Script and Bad USB VBScript 15

5. Implementation 16
5.1 Normal USB Drives as Incredible Bad USB Devices 16
5.2 Selecting Appropriate USB Portable Storage Devices 16
5.3 Installing Custom Firmware 16
5.4 Scripting the Attack 17
5.4.1 Using Ducky Script 17
5.4.2 Using VBScript 19
5.5 Testing and Refining 21
5.6 Deploying the Attack 21

6. Project Implementation 22
7. Conclusion 29
8. References 31

LIST OF TABLES

Table 1. Comparison between Normal USB and Bad USB 12

Table 2. Comparison between Normal USB and Bad USB Ducky Script 13

Table 3. Comparison between Normal USB and Bad USB VBScript 14

Table 4. Comparison between Bad USB Ducky Script and Bad USB VBScript 15

Table 5. VBScript takes over keyboard 28

List of Figures

Figure 1. Ducky Script Example flowchart 18

Figure 2. VBScript Example flowchart 19

Figure 3. Project Code flowchart 27

1. INTRODUCTION

As far as the campaign is concerned, the modern compact media device known as flash disk or USB drive is common equipment for transferring and storing data. Originally they were just simple functions which can be plugged into any device and used without much thought but today they are sophisticated systems used for numerous complicated purposes which have changed the face of security as we know it. On the one hand, the portability of USB drives is indisputable, which is particularly useful when remapped to function as Bad USB devices.

It is concerning to study the newly identified threat associated with Bad USB devices, including the various ways that can be employed due to the Ducky Script and VBScript. Bad USB attacks rely on the fact that USB is a universal technology that is generally trusted by individuals, and is implemented in many host and endpoint devices, thereby turning a conventional USB stick into a highly efficient tool for performing malicious operations on unguarded systems.

It is also important to understand that there is a clear line between normal USB operation and Bad USB assault to fully appreciate the issue at its best. Ordinary USB drives are developed for innocent uses like safe storage of data and connection of peripherals. They act within certain parameters recommended and put into place to optimise compatibility and security. But Bad USB is a device that exploits these very standards to get around security measures threaten to execute unauthorized commands and compromise the system.

For example, it is possible to use such a tool as the USB Rubber Ducky, which applies Ducky Script as a scripting language that, upon activation, captures the victim's whole OS session. As an extension, Ducky Script can perform fake keyboard commands for simulating complex sequences of commands, meaning that attackers can quickly input payloads and remain invisible. Likewise, VBScript, a scrip language that was originally developed by Microsoft, can be used to take advantage of USB autorun opportunities and it becomes even worse for Bad USB attacks.

By using this system-level outline, we will examine how Bad USB devices operate and what capabilities they use in comparison to regular USB drives. This thesis therefore seeks to define the threat from a purpose, appearance, function, security, and system perspective with the intention of presenting asecure thorough description of the threat and its implications in order to recommend suitable detection and mitigation techniques.

In conclusion, these new and improved malware known as Bad USB devices prove that users should be more cautious than ever with these devices and that security should be better enforced. In our modern day and age, it only means that the alertness and preparedness of each one of us must never falter for any chance of Bad USB to penetrate our lives and, on the process, endanger our well-being.

2. Literature Review

2.1 Overview of USB Technology

Universal Serial Bus or USB as commonly referred to has been one of the most important developments in the modern age of computing that enables trendsetting data transfer and device connections. USB was first conceived to simplify the method of connecting peripherals to computers and replace a wide range of various interfaces. USB standard is a universal and manageable hardware three in one connectivity through which data transfer, electrical connection, and computer device control.

2.1.1 Historical Development

USB technology was first developed in the year 1996 and aimed at reducing tremendous connections between computers and related peripheral equipment. For starters, the UK to USA will mean USB 1.0, offered a means of data transfer with speed ranging up to 12 Mbps. Further enhancements are available with the successive revisions namely, the USB 2.0, 3.0, 3.1, and 3.2, have by far improved these speeds than that of the preceding one such as the USB 3. Achieving Transfer Rates up to 20 Gbps of the two-standard achieving.

2.1.2 Functional Components

Physical Connection: USB cables mainly consist of a standard Type-A plug on one of the ends and a plug that is specific to the device the cable is to be connected on the other end. They are provided with connecting ports that should fit securely in order to give the correct and strong connection.

Power Delivery: it is an important feature of USB ports as they provide a power supply to connected devices, eliminating the need for separate power connections. USB 2.0 and its subsequent versions can provide more current, which is crucial for

powering up devices. For instance, a menu at the bottom of the screen might show 5V • 500mA on 0 ports, indicating the current power delivery status.

Data Transfer: USB ports are used to transfer data between devices, and different versions have different transfer rates. USB 2.0 supports up to 480 Mbps. USB 3.0 can operate at speeds of up to 5 Gbps, USB 3.1 supports up to 10 Gbps, and USB 3.2 allows for data transfer speeds ranging from 2 Gbps up to 20 Gbps.

Plug-and-Play: USB themselves allow the connection or removal of devices while the computer is still running; they are hot-swappable. One of the most interesting features of this kind of connection is that it enables hassle free addition and deletion of accessories.

Device Enumeration: When USB device is connected the host device tries to detect and initialize the device through a mechanism which is known as enumeration. Then the operating system provides the device with a unique address then the operating system loads the necessary drivers.

Device Classes: The USB device can be of a particular class, for instance; the class one of the USB devices supports are mass storage, human interface devices (HID), and network adapters. This classification aids the host and other devices in formatting subsequent communications to suit the process.

USB Hubs: These devices offer extensions of the number of USB ports through a number of devices to a single port among the host. Some of them are powered, which in turn can produce more power to other devices that are connected to the hub.

2.2 USB Security Threats

While they are very useful and easy to transport, devices such as USB can compromise the security of a computer or network. USB technology is integrated into almost every platform, which makes it flexible and convenient, but for cyber attackers, this aspect is the technology's biggest weakness. The subsequent sections examine typical threats against USB related security.

Malware Propagation: Malicious USB gadgets are capable of transmitting forms that contain Viruses and other malicious software between systems. Autorun virus – it is a kind of virus appeared as a result of the usage of the USB autorun function – the code on the virus begins to work instantly after the device is connected to the computer.

Data Theft: This is because USB drives pose a great threat as they contain unscrupulous individuals who may act in malicious intent by stealing sensitive information. In particular, malware can be temporarily stored on removable USB drives and used to steal data from secure networks, which attackers can physically connect to the systems by deception.

Bad USB Exploits: It then turns to discussing a more recent type of attack called Bad USB where the firmware of a USB device is manipulated to carry out specific actions. The problem of Bad USB is that it functions in a completely different manner than classical viruses and most of the time, it cannot be easily stopped by any antivirus software.

USB Rubber Ducky: One of the most acknowledged tools, which can be utilized in performing Keystroke Injection Attacks is the USB Rubber Ducky. Cloaked in the appearance of a USB flash drive, it emulates a keyboard and continuously enters code that subverts the targeted computer.

2.3 Bad USB, Ducky Scripts and VBScript

Bad USB

High risk USB attacks include manipulation of the firmware of the USB devices to alter their functionality or capabilities. Should a Bad USB stick be reprogrammed it can effectively launch programs that might result in keystroke injection, network spoofing or data leakage. Infact, such attacks are hard to identify since the USB devices are trusted, unlike the other external gadgets.

Ducky Scripts

Scripts are used to program USB Rubber Ducky Devices, and these scripts are written in Ducky Scripts that are easy to program. These scripts imitate keyboard commands, and since the USB Rubber Ducky, means that it can perform a lot of tasks in quick succession. Affected by scripts may lead to opening new doors for hackers to logging into the system, download malware and other malicious activities.

VBScript

Vbscript stands for (Visual Basic Scripting Edition) it is a Microsoft scripting language which has been derived from Visual Basic language. It is mainly used for running script execution in Windows environments and could be implemented in an HTML to run web page interactions or be programmatically useful for administrative tasks.

2.4 Existing Mitigation Techniques

To counteract the threats posed with the aid of Bad USB and Ducky Scripts, several mitigation techniques had been evolved.

Endpoint Security Solutions

Endpoint protection software can help locate and prevent malicious USB activity. These answers often include capabilities like device manipulate, which restricts USB access primarily based on predefined regulations.

USB Device Authentication

Implementing USB tool authentication guarantees that handiest depended on gadgets can connect to a machine. This can be performed via virtual certificate or hardware-primarily based authentication mechanisms.

User Education

Educating customers about the risks associated with USB gadgets is important. Users should be trained to recognize and avoid ability threats, which include now not connecting unknown USB drives to their computer systems.

Firmware Integrity Checks

Regularly checking the integrity of USB device firmware can assist stumble on unauthorized changes. Firmware integrity exams may be computerized and incorporated into protection tracking structures.

Physical Security Measures

Implementing physical security features, such as locking USB ports or using records blockers, can prevent unauthorized get admission to to USB ports. This reduces the chance of malicious USB devices being linked to steady structures.

3. Methodology

3.1 Normal USB Functionality

Traditional technologies or commonly used gadgets such as the Normal USB or the Universal Serial Bus are critical to contemporary computing as they present a fashionable technique for information exchange and power exchange between devices.

3.2 Bad USB Ducky Scripts

Bad USB attacks use this weakness to trick the device into behaving in a negative manner that could be detrimental to the used equipment or even the system. One of them is the Use-Of-Booting Rubber Ducky which is a USB device that program to take keyboard inputs and perform pre-set scripts. Key components include:

USB Rubber Ducky: A keyboard USB drive is a USB device that is used to type keystrokes into a computer and at the same time it contains payload commands to be performed when connected to the computer.

Ducky Script: USB Rubber Ducky is an attack proxy under the form of a USB device that launches a simple scripting language for defining the keystrokes and their actions. It enables different users to write specific sequences of Instructions and operations to be run in a sequential manner.

3.3 Bad USB VBScript

VBScript attacks can also be used to perform bad USB attacks even though it is a scripting language that originated from Microsoft which is mainly used for windows-based platforms. When it comes to the Bad USB attacks, VBScript works for automating actions or even interacting with the operating system,

which makes it appropriate for destructive purposes. Key aspects of VBScript include:

VBScript Basics: As its name suggests, VBScript is a relatively small language which is quite easy to learn, and shares much syntax with Visual Basic. Python is a programming language that is widely used for developing new programs very fast and also for running scripts for Windows.

Variables and Data Types: In VBScript, there is something known as variable, which entails data, that is, information that can be used in the rest of the script. It is an implicit declaration of the data type which is why it is not a strongly typed language.

Control Structures: Later in this tutorial, you will learn about the loops and conditional statements that VBScript uses to change the program's flow.

Functions and Procedures: Similar to other script languages, VBScript features the ability to use function and procedure to form a script for reuse many times of code.

Interacting with the Environment: Just like most modern programming languages, VBScript has the capability to interface with the operating system, in this case Windows and performs action such as modifying files, registry keys and launching other applications.

Error Handling: It is important for VBScript developers to check for errors and have methods to handle them when they occur at runtime.

3.4 Mechanisms of Bad USB Ducky Scripts

Bad USB Ducky Scripts operate through several mechanisms to compromise systems:

Keyboard Emulation: The USB Rubber Ducky is a keyboard, and since it does not look like it is doing anything, it enters the commands reasonably fast, and the user authentication is bypassed.

Script Execution: Malicious scripts perform a list of actions, meaning that such files have the possibility to use the weaknesses of a system.

Payload Delivery: Scripts can not only download other malware to the victim's machine, but also other forms of malware, like backdoors, or remote access tools.

Social Engineering: One of the well-known techniques is actually the use of tricks from the attacker in making the users connect the USB device.

Firmware Manipulation: Some of the operating features include systemless flashing in which custom firmware is written into USB devices to modifications its functionality to avoid being detected.

Automated Exploitation: Scripts are pre-designed to take advantage of known weaknesses; they decrease the time required for the attack process.

Bypassing Security Controls: Malware books can bypass some security measures, namely, antivirus programs can be defeated by such USB devices.

Data Exfiltration: A script can be designed to pull out and feed critical information from the target system.

3.5 Mechanisms of Bad USB VBScript

Malware in Bad USB VBScript is based on the scripting properties of VBScript to execute multiple values, as soon as the contaminated USB device is plugged into a system. These mechanisms include:

Script Execution: Like Ducky Script, VBScript can also perform certain commands without the need to type them manually. This can easily occur when a USB device that has been tampered with is plugged into the computer; the VBScript can activate scripts that carry out activities with wrong intent.

Payload Delivery: VBScript in turn can contain a script for downloading other malware programs on the victim computer or execute some other code. The above script can be used to make connection to other servers in a network with the aim of downloading unwanted files and execute them in the target computer.

System Manipulation: This means that the same VBScript source code can work in any Windows environment and can create, modify, manipulate, delete files, and modify the settings of the operating system as well as control other programs. They say this capability enables the script to contain a set of possible evil deeds to which it can perform.

Registry Manipulation: One more powerful feature of VBScript is that it can adjust the settings of the Windows registry; it means that this script can change the way the system behaves or even disable some security aspects.

Social Engineering: These are fake dialogs and messages which appear on the users' interface and are written in VBScript; the goal is to intimidate the user into performing specific actions which put his computer at risk.

4. Comparative Analysis of Normal USB and Bad USB

4.1 Comparison between Normal USB and Bad USB:

Feature	Normal USB	Bad USB
Purpose	Data storage and transfer	Exploit vulnerabilities or execute malicious actions
Appearance	Standard USB flash drive	Disguised as various devices
Functionality	Performs intended data transfer functions	Executes unauthorized commands or payloads
Security	Generally secure	Bypasses security measures
Manufacture	Reputable manufacturers	Created or modified by malicious actors
Usage	Legitimate purposes	Cyber attacks or penetration testing
Detection	Can be scanned for viruses/malware	Harder to detect due to disguised nature and malicious intent
Protection	Best practices include using trusted sources	Requires security measures like endpoint protection, network segmentation, and user education
Impact on System	Generally benign unless infected	Can lead to data theft, system compromise, or disruption of operations

4.2 Comparison between Normal USB and Bad USB Ducky Script:

Feature	Normal USB	Bad USB Ducky Script
Purpose	Data storage and transfer	Perform automated keyboard-based attacks
Appearance	Standard USB flash drive	Often disguised as a normal USB device
Functionality	Performs intended data transfer functions	Emulates a keyboard to send predefined keystrokes
Security	Generally secure	Bypasses security measures to execute commands
Manufacture	Reputable manufacturers	Produced by entities for penetration testing or malicious use
Usage	Legitimate purposes	Used in cyber attacks or security testing
Detection	Can be scanned for viruses/malware	Difficult to detect due to keyboard emulation
Protection	Best practices include using trusted sources	Requires endpoint protection and user education
Impact on System	Generally benign unless infected	Can lead to unauthorized command execution and system control

4.3 Comparison between Normal USB and Bad USB VBScript:

Feature	Normal USB	Bad USB VBScript
Purpose	Data storage and transfer	Automate tasks or execute malicious scripts
Appearance	Standard USB flash drive	Often disguised as a normal USB device
Functionality	Performs intended data transfer functions	Executes VBScript to perform tasks or exploit vulnerabilities
Security	Generally secure	Bypasses security measures to run scripts
Manufacture	Reputable manufacturers	Created or modified for automation or malicious purposes
Usage	Legitimate purposes	Used in cyber attacks or automation tasks
Detection	Can be scanned for viruses/malware	Hard to detect scripts running upon connection
Protection	Best practices include using trusted sources	Requires endpoint protection and user education
Impact on System	Generally benign unless infected	Can lead to automated task execution or exploitation

4.4 Comparison between Bad USB Ducky Script and Bad USB VBScript:

Feature	Bad USB Ducky Script	Bad USB VBScript
Purpose	Perform automated keyboard-based attacks	Automate tasks or execute malicious scripts
Appearance	Ducky Script	VBScript
Functionality	Emulates keyboard to send keystrokes	Runs VBScript upon connection
Security	Automated command execution via keystrokes	Script execution for task automation or exploitation
Manufacture	Difficult to detect due to keyboard emulation	Hard to detect scripts running upon connection
Usage	Typically simple commands and keystrokes	Can include complex scripting and automation
Detection	Quick execution of predefined commands	Versatile scripting for various attack vectors
Protection	Limited to keystrokes and simple commands	Extensive scripting capabilities for varied tasks
Impact on System	Can lead to unauthorized command execution and system control	Can lead to automated task execution or exploitation

5. Implementation

5.1 Normal USB Drives as Incredible Bad USB Devices

When creating a Bad USB device, a normal USB drive is first altered by having its firmware changed, such that it running scripts as soon as it is connected. This often entails configuring the USB to operate as a keyboard or some other hardware input, so that it can provide commands pre-prepared by the hacker swiftly and involuntarily on the part of the PC proprietor.

5.2 Selecting Appropriate USB Portable Storage Devices

Choosing an appropriate USB drive is vital in the making of a Bad USB gadget since not all of them are appropriate for this function. Portability and versatility have made microcontroller-based USB drive, for example the ATtiny85 a common feature due to their programmable nature. These drives also provide support for firmware upgrades therefore making them suitable for transforming into malicious devices.

5.3 Installing Custom Firmware

Instead of that, the USB drive needs to be formatted with new firmware that is specific to its purpose. This firmware emulates the USB device and makes the device to emulate the keyboard function such that it can type for itself fully. An example is the “Twin Duck” firmware used with USB Rubber Ducky devices, which allows the USB storage device also act like USB keyboard.

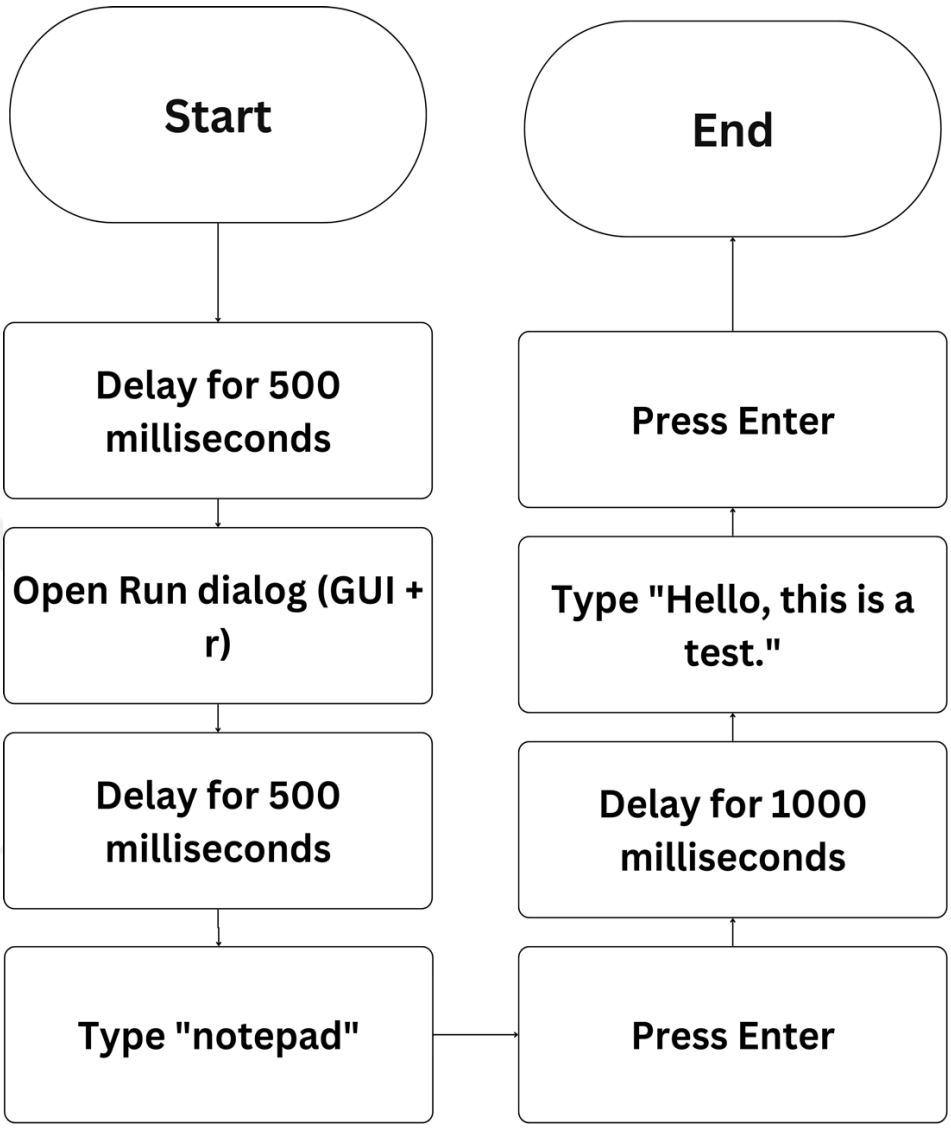
5.4 Scripting the Attack

5.4.1 Using Ducky Script

USB Rubber Ducky devices are pre-programmed keystroke injection tool while Ducky Script is the script language that works with it. It lets the keystrokes and delays to be set as well as commands when USB device is connected to the computer system.

Ducky Script Example:

```
REM This script opens Notepad and types a message
DELAY 500
GUI r
DELAY 500
STRING notepad
ENTER
DELAY 1000
STRING Hello, this is a test.
ENTER
```



- REM: Adds comments in the script.
- DELAY: Introduces a delay to ensure commands execute properly.
- GUI r: Simulates pressing the Windows key + 'r' to open the Run dialog.
- STRING: Types the specified text.
- ENTER: Simulates pressing the Enter key.

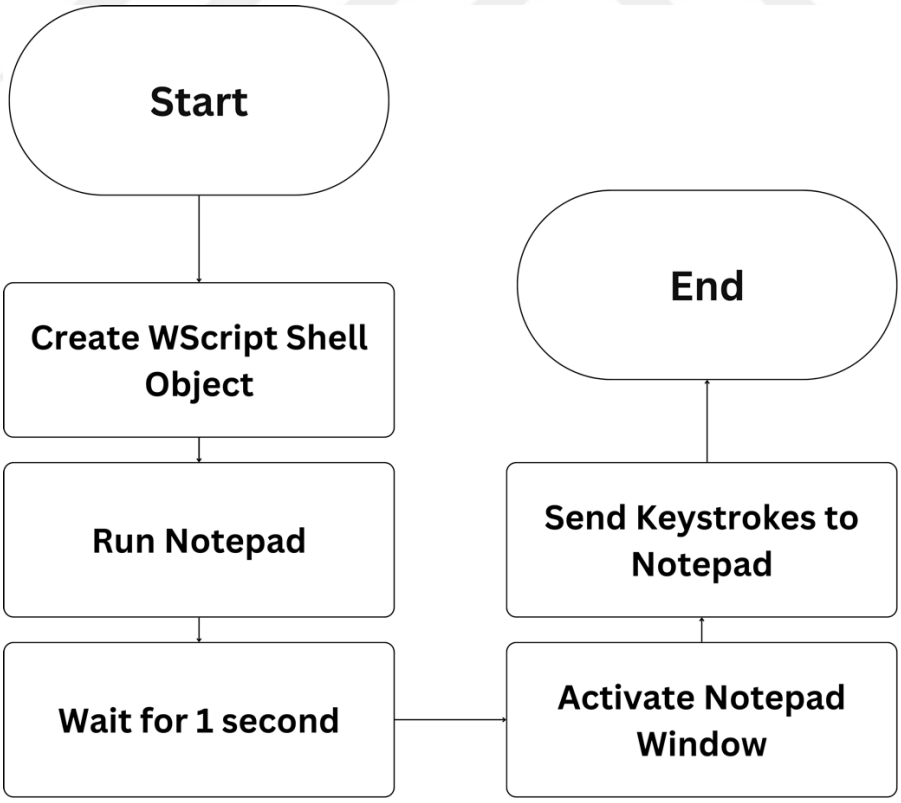
This script opens the Run dialog, types "notepad" to open Notepad, and then types "Hello, this is a test."

5.4.2 Using VBScript

VBScript (Visual Basic Scripting Edition) is a scripting language developed by Microsoft, primarily used for automation of tasks in Windows environments.

VBScript Example:

```
Set objShell = CreateObject("WScript.Shell")
objShell.Run "notepad.exe"
WScript.Sleep 1000
objShell.AppActivate "Untitled - Notepad"
objShell.SendKeys "Hello, this is a test."
```



- Set objShell = CreateObject("WScript.Shell"): Creates an instance of the WScript.Shell object.
- objShell.Run "notepad.exe": Opens Notepad.
- WScript.Sleep 1000: Pauses for a second to allow Notepad to open.
- objShell.AppActivate "Untitled - Notepad": Activates the Notepad window.
- objShell.SendKeys "Hello, this is a test.": Sends keystrokes to Notepad to type the message.

This script opens Notepad and types "Hello, this is a test."

5.5 Testing and Refining

Each of these creations should work as a predefined attack permits testing is an essential phase to ascertain that the scripted attacks operate properly on different operating systems and platforms. This involves:

Compatibility Testing: The script works on the various windows: The last consideration is that they should work on diverse windows version.

Refinement: Configuration changes in the delay and the number of keystrokes it took to input and execute a command due to differences in systems.

5.6 Deploying the Attack

A Bad USB attack functions through delivering malevolent software to a target computer through the physical connection of the USB port. The primary method of infecting the targeted device is accomplished with the use of social engineering, where the target is forced to insert the device, for instance by presenting it as a normal USB flash memory stick or simply using curiosity by labeling the stick.

6. Project Implementation

Here am going to be explaining all the steps I've made to make this project:

- The website that helped with the code:
  https://ss64.com/vb/

- The code I wrote to make the program work:

```
set x=createobject("wscript.shell")

x.run "Cmd"
wscript.sleep 500
x.sendkeys "shutdown /s"
wscript.sleep 500
x.sendkeys "{ENTER}"
wscript.sleep 3000
x.sendkeys "{ENTER}"
wscript.sleep 500
x.sendkeys "exit"
wscript.sleep 500
x.sendkeys "{ENTER}"
wscript.sleep 1000

x.run "notepad.exe"
wscript.sleep 1000
x.sendkeys "You have been hacked !!!"
wscript.sleep 1000
x.sendkeys "^w"
wscript.sleep 1
x.sendkeys "{RIGHT}"
wscript.sleep 1
x.sendkeys "{ENTER}"
```

- The code explanation:

    - **set x=createobject("wscript. shell")**

    This statement kicks off the creation of a "WScript.Shell" object, which is required for interacting with Windows Shell environment, and living element. The instantiation of such kind of object will let modify the system-level functionalities.

    - **x. run "Cmd"**

    This line creates the shell object. The cmd. exe, which is the main method of the system, will be invoked. Opening a command line interface in this process, which executes further commands.

    - **wscript. sleep 500**

    A pause of brief duration is applied, so that the system reacts, and the execution of commands is done sequentially. It also prevents any possible conflict or error due to a quick command just being produced.

    - **x.sendkeys "shutdown /s"**

    The top line of the Command Prompt lets you send the shutdown /s command using the SendKeys method, which tells the system to initiate an immediate shutdown sequence. These instructions show system level command which include fatal consequences for its operation if it will not be done with the user's permission or authorization.

    - **wscript. sleep 500**

    Yet another pause takes place in order give enough time for the system to gracefully execute the immediate command before script continues.

- **x. sendkeys "{ENTER}"**

As the SendKeys function fires the Enter keystroke that it simulated, the Command Prompt, where the shutdown takes place, is being reaffirmed to perform that action.

- **wscript. sleep 3000**

As the shutdown sequence progresses, a much longer delay is set to manage all the systems that may be inactive or see notifications, such as those related to other software applications.

- **x. sendkeys "{ENTER}"**

An additional Enter key will follow, and if there are any system prompts or simulations that may appear during the shutdown, then this could deal with them subsequently.

- **wscript. sleep 500**

In order to maintain a specific speed and avoid execution conflicts, a short time period is given. With this approach, the play will proceed gradually and without stoppages.

- **x. sendkeys "exit"**

This line calls SendKeys command line and exits Command Prompt or all command prompt sessions.

- **wscript.sleep 500**

Yet another pause takes place in order give enough time for the system to gracefully execute the immediate command before script continues.

- **x. sendkeys "{ENTER}"**

The last action is to enter the simulated keystroke of Enter button in Command Prompt to send the closure command and successfully close the Command Prompt session.

- **wscript. sleep 1000**

A short gap appears in the process to let the OS to start the Notepad application if necessary which is shown by further actions being performed.

- **x.sendkeys "You have been Hacked !!!"**

Input "You have been hacked !!!" to the newly opened Notepad window with SendKeys method to show how the intruder could do scripting or manipulated the operating system.

- **wscript. sleep 1000**

Another the statement of expression made in pause is used which further allow the language delay to be taken before the script move onto next that enable the smooth running sequence of script .

- **x. sendkeys "^w"**

This sentence addresses the keystrokes that are sent over the Ctrl + W to the Notepad window as an example of the user exiting the active document.

- **wscript. sleep 1**

The script is halted for a moment so the code to advance its execution smoothly and adhering to good coding techniques and readability norms.

- **x. sendkeys "{RIGHT}"**

A series of hypothetical characters is afterwards transmitted, that is the arrow keys corresponding to the Windwos Notepad document windows that may hold some prompt boxes or dialogues.

- **wscript. sleep 1**

  The script likewise includes an intermission into the pause which provides more readable script and the subsequent commands can be done properly.

- **x. sendkeys "{ENTER}"**

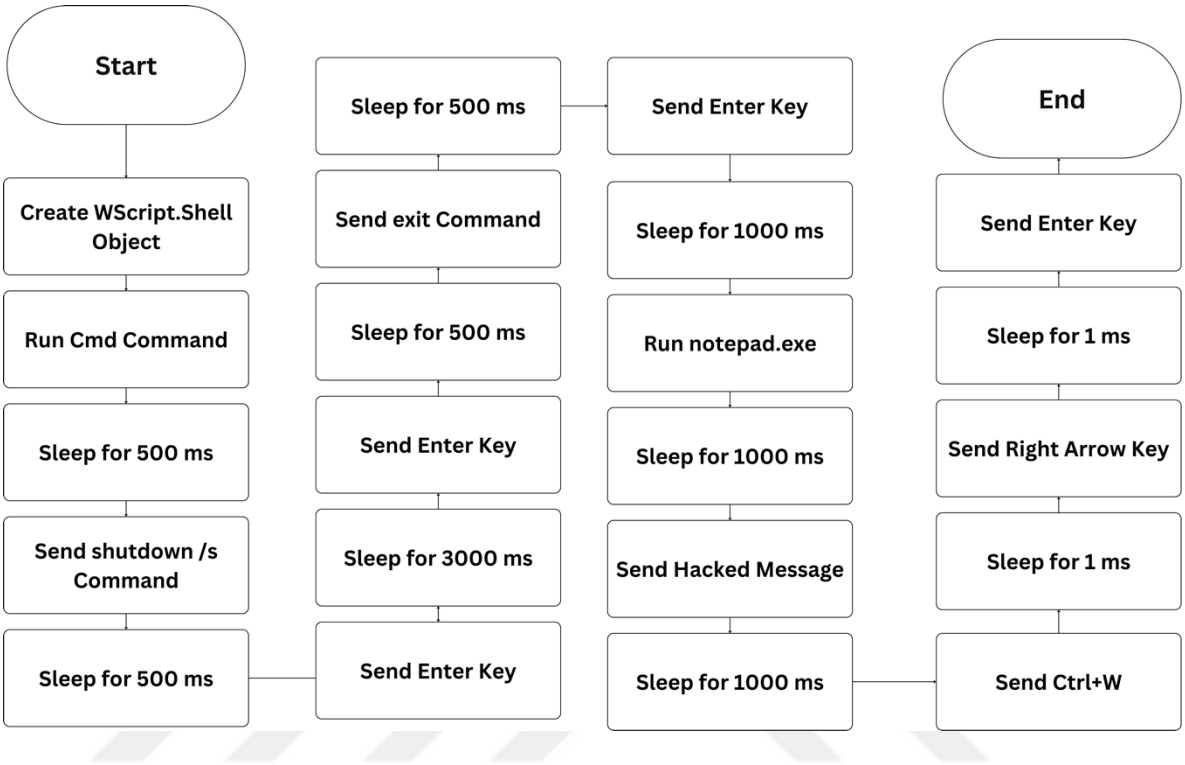
  And at last, an Enter keypress is broadcast and consequently solutions of any user inputs and system prompts will be processed in the run.

Overall, this VBScript file executes a series of commands:

- Opens the command prompt.
- Shuts down the system.
- Exits the command prompt.
- Opens Notepad.
- Displays a message in Notepad.
- Closes Notepad.

Additionally, it attempts to handle any dialogues or confirmations that may appear during the process.

Project code flowchart:



- **USB AutoRun Creator software**

    - Download and Install USB AutoRun Creator
    - Launch the USB AutoRun Creator
    - Insert Your USB Drive
    - Create a New Project
    - Add Files to the USB Drive
    - Set AutoRun Actions
    - Save and Build the Project
    - Test the USB Drive
    - Finalize and Use

The autorun for the project works as long as the autorun blocker is disabled in the device that the USB is connected to.

- VBScript takes over keyboard

here is a table that explains how VBScript takes over keyboard:

Key/Character	SendKey	Description
~	{~}	Send a tilde (~)
!	{!}	Send an exclamation point (!)
^	{^}	Send a caret (^)
+	{+}	Send a plus sign (+)
Backspace	{BACKSPACE} or {BKSP} or {BS}	Send a Backspace keystroke
Break	{BREAK}	Send a Break keystroke
Caps Lock	{CAPSLOCK}	Press the Caps Lock Key (toggle on or off)
Clear	{CLEAR}	Clear the field
Delete	{DELETE} or {DEL}	Send a Delete keystroke
Insert	{INSERT} or {INS}	Send an Insert keystroke
Cursor control arrows	{LEFT} / {RIGHT} / {UP} / {DOWN}	Send a Left/Right/Up/Down Arrow
End	{END}	Send an End keystroke
Enter	{ENTER} or ~	Send an Enter keystroke
Escape	{ESCAPE}	Send an Esc keystroke
F1 through F16	{F1} through {F16}	Send a Function keystroke
Help	{HELP}	Send a Help keystroke
Home	{HOME}	Send a Home keystroke
Numlock	{NUMLOCK}	Send a Num Lock keystroke
Page Down	{PGDN}	Send a Page Down or Page Up keystroke
Page Up	{PGUP}	Send a Page Down or Page Up keystroke
Print Screen	{PRTSC}	Send a Print Screen keystroke
Scroll lock	{SCROLLLOCK}	Press the Scroll lock Key (toggle on or off)
TAB	{TAB}	Send a TAB keystroke

To specify keys combined with any combination of SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following:

For SHIFT prefix with **+**
For CTRL prefix with **^**
For ALT prefix with **%**

7. Conclusion

From this thesis, we discussed the various security challenges that relate to Bad USB devices, especially the one that utilizes Ducky Script and VBScript. The analysis helped to reveal dramatic differences between the typical, regular operation and Bad USB operation and revealed the general knowledge about the hazardous USB devices.

Key Findings:

Flexibility of USB Technology: Although USB technology is indeed perhaps the greatest invention of the century for data transfer and peripheral support the flexibility is this technology is both strength and weakness at the same time. It targets the reliability and pervasiveness of USB technology in executing unwanted commands and implementing malicious code in systems that should otherwise be secure.

Mechanisms of Attack: USB devices can be maliciously programmed with Ducky Script and VBScript is a significant part of USB programming. For USB Rubber Ducky devices, there is Ducky Script that together with a keyboard emulation makes it possible for a hacker to simulate keystrokes and even any complicated string of commands. VBScript is inherently tied to Windows' scripting layer, thereby enabling a wide range of malicious operations to occur seamlessly behind the scene.

Transformation Process: They are created by fitting a normal USB drive with a new firmware, and scripting and rigorously testing it closely. This process highlights the plausibility of an adversary to leverage conventional and functional USB peripherals for immoral intentions In this line of thought, identification remains paramount and challenging at the same time.

Mitigation Techniques: Some of the measures that can help in reducing the risks of Bad USB devices include endpoint security products, USB device identification, training users, scanning firmware integrity and other security measures that imp Pro/Con error t against physical access. These are important measures that would help to formulate a good defense against the increasing tendencies of getting attacked through the USBs.

Implications and Recommendations:

The threat posed by Bad USB devices has become a constant risk and implies the need for improved security surrounding USB products. Many organizations and individuals have been victims of these types of attacks thus both parties have to undergo numerous security precautions. These phases comprise applying various technical measures, promoting organizational culture and raising organizational cybersecurity awareness, and outlining the available threats and countermeasures or the best security practices.

To summarize, the development of the USB has been beneficial in terms of making devices portable and easy to use; however, new threats have appeared in relation to this technology. This paper aims to enlighten the community about Bad USB attacks highlighting those mechanisms, consequences, and countermeasures that can be efficiently implemented to protect our environments. More future studies must be directed toward the identification of hope CBRN threat detection techniques as well as countermeasures for the future future threats.

References

- Anderson, R., & Kuhn, M. G. (1997). Low Cost Attacks on Tamper Resistant Devices. In Security Protocols (pp. 125-136). Springer, Berlin, Heidelberg. Access address: https://link.springer.com/chapter/10.1007/bfb0028165

- Conti, M., Dragoni, N., & Lesyk, V. (2016). A Survey of Man In The Middle Attacks. IEEE Communications Surveys & Tutorials, 18, 2027-2051. Access address:https://doi.org/10.1109/COMST.2016.2548426

- Shafique, U., & Zahur, S. (2019). Towards Protection Against a USB Device Whose Firmware Has Been Compromised or Turned as 'BadUSB'. Lecture Notes in Networks and Systems. Access address:https://doi.org/10.1007/978-3-030-12385-7_66

- Stergiopoulos, G., & Gritzalis, D. (2015). Hacking and Penetration Testing with Low Power Devices. Comput. Secur., 49, 274-275. Access address: https://doi.org/10.1016/J.COSE.2015.01.001

- Neuner, S., Voyiatzis, A., Fotopoulos, S., Mulliner, C., & Weippl, E. (2018). USBlock: Blocking USB-Based Keypress Injection Attacks., 278-295. Access address: https://doi.org/10.1007/978-3-319-95729-6_18

- E, S., S, M., & K, S. (2023). Bad USB as HID and it's Mitigations. International Journal for Research in Applied Science and Engineering Technology. Access address: https://doi.org/10.22214/ijraset.2023.51045

- Nissim, N., Yahalom, R., & Elovici, Y. (2017). USB-based attacks. Comput. Secur., 70, 675-688. Access address: https://doi.org/10.1016/j.cose.2017.08.002

- Cannoles, B., & Ghafarian, A. (2017). Hacking Experiment Using USB Rubber Ducky Scripting. Journal on Systemics, Cybernetics and Informatics, 15, 66-71. Access address: https://www.iiisci.org/Journal/pdv/sci/pdfs/ZA340MX17.pdf

- Harianto, H., & Gunawan, D. (2019). Wi-Fi password stealing program using USB rubber ducky. TELKOMNIKA (Telecommunication Computing Electronics and Control). Access address: https://doi.org/10.12928/TELKOMNIKA.V17I2.11775.

- Jeevanantham, M., Kani, C., Visweswaran, N., & Deepalakshmi, P. (2019). Design of Integrated Exploitation Console using Hak5. Access address: https://doi.org/10.35940/ijeat.a1112.1291s419.

- Al-Zarouni, M. (2006). The reality of risks from consented use of USB devices. Access address: https://doi.org/10.4225/75/57B6543434762.

- Fournier, G., Matoussowsky, P., & Cotret, P. (2016). Hit the KeyJack: stealing data from your daily wireless devices incognito. *ArXiv*, abs/1610.05212. Access address: https://www.semanticscholar.org/paper/Hit-the-KeyJack%3A-stealing-data-from-your-daily-Fournier-Matoussowsky/1ab2e717dd1d989442f0e0561dd9b53d2ab7ad0e

- Jeong, H., Choi, Y., Jeon, W., Yang, F., Lee, Y., Kim, S., & Won, D. (2007). Vulnerability analysis of secure USB flash drives. 2007 IEEE International Workshop on Memory Technology, Design and Testing, 61-64. Access address: https://doi.org/10.1109/MTDT.2007.4547620.

- Tischer, M., Durumeric, Z., Bursztein, E., & Bailey, M. (2017). The Danger of USB Drives. *IEEE Security & Privacy, 15*, 62-69. Access address: https://www.semanticscholar.org/paper/The-Danger-of-USB-Drives-Tischer-Durumeric/89e37c24a85b395fa89ea435e815a63340cf8fe6

- Stokes, J. W., Agrawal, R., & McDonald, G. (2020). Detection of Malicious Vbscript Using Static and Dynamic Analysis with Recurrent Deep Learning. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 2020-May, pp. 2887–2891). Institute of Electrical and Electronics Engineers Inc. Access address: https://doi.org/10.1109/ICASSP40776.2020.9054390

- Young, N., & Drees, R. (2018). Cyber security for automatic test equipment. *IEEE Instrumentation and Measurement Magazine*, *21*(4), 4–8. Access address: https://doi.org/10.1109/MIM.2018.8423738

- Farhi, N., Nissim, N., & Elovici, Y. (2019). Malboard: A novel user keystroke impersonation attack and trusted detection framework based on side-channel analysis. *Computers and Security*, *85*, 240–269. Access address: https://doi.org/10.1016/j.cose.2019.05.008

- Monrose, F., & Rubin, A. (1997). Authentication via keystroke dynamics. In *Proceedings of the ACM Conference on Computer and Communications Security* (pp. 48–56). ACM. *Access address:* https://doi.org/10.1145/266420.266434

- Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, *22*, 113–122. Access address: https://doi.org/10.1016/j.jisa.2014.09.005

- Humphries, S. (2024, May 4). USB explained: All the different types (and what they're used for). *How-To Geek*. Access address: https://www.howtogeek.com/53587/usb-explained-all-the-different-types-and-what-theyre-used-for/