An Exploration of Rubber Ducky Development

Introduction

A Rubber Ducky (RD) is a hardware device used for penetration testing in cyber-security. When plugged into a system, the RD will inject a payload of keyboard inputs into the system to interact, attack and extract information from the target system. This report details my learning process and development of an RD using a Raspberry Pi Pico W from scratch and in the process, learning about what is required to construct a strong attack using a RD. In the Payloads section, I categorise attacks based on each of the foundational aspects of an RD attack, interactions, attacks, obfuscation and exfiltration. Going into each payload, a brief summary and thought process will be provided as well as an analysis of how to defend against the payload, finalised by any interesting learnings or additional notes. The majority of the language is written in Arduino IDE, with an additional use of PowerShell scripting (C#).

Payloads

Interaction

My initial thoughts going into the development of this RD was how would I interact with the system in the first place? From fundamental researching and comparison to commercialised pen-testing equipment such as the Hak-5 RD it revealed that the majority of interactions were done by emulating a Keyboard using a microcontroller such as the Raspberry Pico W. As such, the following payloads use the Pico W were developed to test and improve my skills in utilising keyboard interacts to navigate the system and elevate my permissions as an attacker.

Key Press Payload

Summary: This payload is designed to test the keyboard interaction of the RD with the system. The payload triggers the “Windows + D” hotkey to close all windows and force the computer to exit to the desktop.

Implications: Attackers will be able to type and freely inject any sort of keystrokes into the system. Access to all windows hotkeys should be explored further as a method to navigate the system.

Defence: Since the RD is emulated as a keyboard device on the system, it can be difficult to guard against it using regular anti-virus and windows defender. As such, defenders can install USB-authorization software to force serial ports to whitelist themselves. However, to ensure that no form of communication is possible between the RD and the system, USB ports can be disabled in the BIOS configuration menu by setting them to “No Boot”, although this is incredibly impractical as all serial ports must be disabled to be a reliable defence, inconveniencing the defender.

Additional Notes and Learnings:

1. Defending against the initial attack of a rubber ducky means the system is physically compromised, the majority of online resources recommend simply not letting your device be unkept to be open to a RD attack. Digital hygiene and security is not only involved in firmware, but ensuring that your computer is in a safe physical space is just as important.
2. The RD does not have any sort of feedback loop, it simply emulates a keyboard and payloads guess what is on the screen by trying to force environments using hotkeys. As such, navigation improvements can be made by keeping track of what processes are running and storing these in PowerShell variables or a text file for later use.

File Delete Payload

Summary: This payload demonstrates how an RD can be used to elevate the permissions of an attacker and delete any file using the Windows PowerShell by deleting a text file on the Desktop called Dummy.txt.

Implications: Attackers being able to delete information means that critical data and infrastructure can be deleted by the attacker. By deleting these files, Attackers can disrupt a users access to their system or delete traces of any scripts or applications used by the attacker. Furthermore, the escalation of permissions to an administrator unlocks all capabilities of the PowerShell, leaving the system open to all PowerShell scripting attacks.

Defence: This payload can be mitigated using user intervention. The PowerShell script can be interrupted by hitting any key to change the script, for example hitting the enter key while the script is typing is likely to break the

payload entirely. Furthermore, closing the PowerShell environment while the script is executing will also render the attack entirely useless.

Additional Notes and Learnings:

1. As the RD does not have any sort of feedback from the system, it is essentially blind as to what is on the screen. It seems the Windows + D hotkey used to force the user to the desktop actually does not close Picture-in-Picture environments, which can interrupt with the script.
2. User interruptions for the Payload can be applied to all later payloads and should be assumed. Although, it should be noted that RD execution time can become incredibly fast, such that scripts cannot be interrupted manually in a reliable fashion, but in my demonstration, scripts will be slowed down to properly show how commands are written. Even with this optimisation, methods of hiding executables and scripts will be explored further in later payloads.

NotePad Payload

Summary: The NotePad payload demonstrates how scripts can be written within a users system using the NotePad application and saved to any desired user directory for later use. The example, writes “Hello World!”, into a NotePad file and saves it to the user typically restricted documents folder.

Implications: Attackers will be able to write scripts and save them within a users files. This allows Attackers greater flexibility in what they are able to execute using the RD’s limited storage space. This payload demonstrates a complete manipulation and navigation of the System’s File infrastructure.

Defence: As this payload is a combination of previous payloads, refer to previously mentioned payloads for viable defence options.

Attack

The following payloads demonstrate the ‘attack’ capabilities of my RD device. Considering interaction and privilege escalation of the attacker is now well achieved, the following payloads will consider methods to extract information, force unintended actions or reduce the defences of a defender.

Firewall Payload

Summary: The Firewall payload uses the Windows PowerShell to disable the all Windows Firewall Protections.

Implications: Disabling the Windows firewall protections allows the attacker to download malicious files onto the system without Window's asking for permissions, simplifying attack lines as malware, ransomware and any other executable can now be downloaded easily. Further, disabling the Firewall will also disable any network protections Windows has in place, the system can now freely communicate with any servers.

Defence: As the user has administrator privileges. Any sort of software used to whitelist or restrict access to the Firewall settings will be generally ineffective. Thus, to defend against such a payload, any form of physical interruption is required to stop the PowerShell script from running. Changing to a different window while the commands run.

Additional Notes and Learnings:

1. Along with the Firewall, Windows Defender can also be uninstalled as an administrator, using the following command.

```
Uninstall-WindowsFeature -Name Windows-Defender
```

Screenshot Payload

Summary: In order to demonstrate how RD payloads can capture any information on the screen, a payload which executes the

Implications: By taking screenshots, the attacker can technically capture any information on the screen at any time.

Defence: Disabling the "use print screen to open screen capture" setting in the Windows User accessibility settings will render this payload ineffective. Furthermore, the RD's reliance on hotkeys such as Win + D to navigate the system can be defended against by disabling ALL windows hotkeys in the config settings. However, the defender will be sacrificing a large portion of convenience which hotkeys bring when disabling.

Additional Notes and Learnings:

1. If multiple monitors are used, print screen will still capture all three monitors in an extremely stretched image.

YouTube Payload

Summary: This payload uses the command prompt to open a link in the default browser of the system. In the example, it opens to a funny YouTube video.

Implications: Being able to open any link is incredibly useful for an attacker. Malicious sites such as phishing links and websites which contain XSS attacks.

Defence: Disabling the run or command prompt will stop the payload from performing any interactions. As Run and command prompt are not often utilised, it is a viable defence against such payloads, although again convenience of the users capabilities to interact with the system is being sacrificed for additional security. Otherwise, the internet can be disabled to stop any sites from being reached, however again this severely impedes the users defenders capability to even use their computer themselves.

Additional Notes and Learnings:

1. This payload actually can execute without the use of the command prompt. The Run program is sufficient in opening links by simply typing the link in directly.

Shutdown Payload

Summary: The shutdown payload utilises the Windows quick access menu with regular keyboard inputs to shut down the target computer.

Implications: An attacker's capability to shutdown a computer can be used as a blanket to cover up any sort of malicious applications or scripts written, as applications will be closed on shut down.

Defence: Defending against this payload would involve disabling the windows key itself, which can be done using the registry editor. Outside of this, the payload is generally difficult to defend against and the previously mentioned physical defences or disabling USB ports should be considered.

Wi-fi Password Steal Payload

Summary: In the search for useful system information, this payload utilises PowerShell commands to extract a users network name and password which they are connected to. The payload can also be modified to extract the information of ALL previously connected networks.

Implications: By accessing and saving the Wi-Fi information, attackers able to exfiltrate the information to increase their attack surface. With network access, attackers can then intercept traffic to gain information such as login details, messages or files.

Defence: Not being connected to the internet using Wi-Fi is an obvious defence for this payload, as the script will no longer execute properly. However,

if enabled to extract information for all previously connected networks, this defence will not be viable. Thus, disabling the PowerShell or command prompt is the best option to defend against this payload.

Hide

In the process of developing these payloads, it is clear that user intervention can have a major effect on payload effectiveness. Thus, I have chosen to explore how to hide the processes which occur in the RD such that users don't notice the attack occurring in the first place making defenders less likely to intervene.

Hidden Download Payload

Summary: A payload designed to minimise and hide processes while they run in the background and also demonstrate how processes can be killed. In the example, PowerShell is used to install the Spotify application to the desktop without actually showing the background process of loading the installer itself.

Implications: The ability to download software and execute is an enormously dangerous implication for an attacker. Attackers can download any form of malicious software such as malware, Trojans, spyware and ransomware to enhance their remote attack capabilities after the payload is complete. Assuming that PowerShell scripts can be run in an optimised and time efficient fashion, such that defender intervention is not viable, executables can be run on a system in the background, entirely hidden from the user interface.

Defence: This payload heavily relies on the use of the Windows PowerShell and an internet connection. Disconnecting from the internet would be a strong defence, as no software would be able to be downloaded, however, this is an impractical assumption and is unlikely to be used. Disabling the PowerShell can also be done, however can also break almost all executables which rely on PowerShell to install applications as well as Windows infrastructure such as the System Center Config Manager used for essential patches and updates and Start-up scripts used for checking system statuses and performing maintenance. In addition, Detection of background applications can be done manually, using the Windows Process manager.

Additional Notes and Learnings:

1. These defence methods can apply to all later Payloads which rely on the PowerShell, but is generally considered impractical as it will disable all

installers and break windows interactions, as such it will not be considered further.

2. Some files downloaded using the PowerShell requires elevated user permissions, as it must be confirmed by an administrator. Further payloads will explore this and the implications of such escalation.
3. Windows defender will stop the download of unlicensed or unrecognised software. Further payloads will explore how this can be disabled.
4. Running PowerShell scripts in theory, should be possible with the PowerShell itself minimised. However, the PowerShell itself seems to not be able to be hidden from the sidebar. An alternative to be further explored would be masking or hiding the PowerShell window either by changing the icon, moving the window itself or somehow executing PowerShell scripts without the application open.
5. It seems some processes cannot be hidden using PowerShell scripts, attaining the setup.exe is possible to run in a hidden manner, but the executable itself seems to always pop-up.

Exfiltrate

Now that the RD has capability to both interact with the system and gather useful information. This information must be exfiltrated from the system to the attacker. The following payloads explore two common techniques using email and a remotely hosted cloud storage.

Email Exfil Payload

Summary: Sends an email to the attacker with the desired file using the PowerShell.

Implications: Any file size less than 25mb will be allowed to attach to an email. Thus, the attacker will be able to send massive text files of information across systems. Furthermore, disabling web browser security will improve the capability of the attacker to download malicious files from the internet without the use of PowerShell.

Defence: To defend against this payload it is quite simple. Adding 2FA will automatically make Web Browser security impossible to disable without a secondary authentication. Additionally, any form of keypress such as tab, escape or switching tabs in the web browser, will interrupt the payload, as it

uses manual keystrokes to navigate what is selected in the browser in order to disable security measures.

Additional Notes and Learnings:

1. **This payload is NO LONGER supported from Chrome, as the functionality to enable “less secure app access” no longer exists. Alternative payloads should be considered.**
2. If your computer is controlled by an external organisation or workplace, web browser security CANNOT be disabled without an external administrators approval, rendering this payload useless.
3. This payload assumes that FireFox is the default browser. However, in combination with previous payloads this can be forced using PowerShell commands to install FireFox in the background and set it as a default browser.

Remote Access Payload

Summary: This payload uploads a target file from the users desktop to a remote storage server using SFTP cloud connections. This connection and file transfer is achieved using PowerShell scripting. This payload is an extension of the Wi-fi password steal as it is used in combination with it to exfiltrate the information.

Implications: The capability for the attacker to send files of any size to a remote cloud storage will better cover the tracks of the attacker and offers an alternative exfiltration method if Email is unreliable. Sensitive information gathered by the attacker such as the text file containing network information (Wi-Fi Payload) can be sent back to the attacker even when they are not in range of the RD device.

Defence: A network filter can be implemented to restrict outgoing SFTP connections to only trusted servers. Otherwise, interruption of the file transfer can be done using the escape key or closing the PowerShell, which is viable as file transfer for even small text files takes up to 10 seconds to completely transfer.

Additional Notes and Learnings:

1. This payload assumes that the previous file download and hide process payload is used to install WinSCP to allow for easy SFTP cloud connection. Although this can be achieved manually through PowerShell scripting the

process is quite arduous. Otherwise, WinSCP can be downloaded by manually traversing the website using Tab and Enter, however this payload is prone to fault and defender interruption.

Difficulties and How I overcame them.

- **Time Management**
    - Time management was a huge factor when developing an open ended project like this for the first time outside of the workplace. While doing it alone, balancing it with all the other activities was an extreme challenge as I am currently working two jobs simultaneously alongside university work. I overcame this extreme challenge but setting realistic expectations for myself and staying as open as possible to tutor feedback and asking for assistance when needed. I believe this course both invoked a great deal of stress, on myself but pushed me to greater heights of organisation and workflow. Consistently asking for help for my SAP, and communicating frequently with my tutor to clarify what the most effective and time efficient techniques were for applying the theory learned in lectures has taught me that within reason, a question or idea is never stupid, just hasn't hit the correct audience yet.
    - Furthermore, due to the broad scope and massive workload of this project it really pushed me to prioritize the tasks which I believed to be the most important, whether it be health, learning or relationships. To really prioritize my understanding and development over results. Something I will definitely be bringing into my future at university and the workplace.

- **Project Direction and Moving the Goal Post**
    - During the SAP, I really struggled with setting a solid project direction. The overwhelming selection of pen-testing equipment (Wifi-pineapple, Rubber Ducky, RFID spoofer) made it difficult to choose and gave me an extremely broad scope to research initially. To overcome this, I actually spent as much time as possible (including spare time), to expose myself to entertaining content whether it be forums or videos of each piece of equipment to decide which I found the most interesting.

    - One thing I did notice, with a single person project, I found myself moving my goals further and further away to push my learning even deeper. This spread me far too thin at times, researching deep topics such as exfiltration techniques without Wi-Fi connection and RFID cloning in preparation for a section which I couldn't even reach in the end. From this, I learned that refining what you have first to achieve clearly defined goals is the most effective use of my time and energy. A vague understanding of many topics does very little for my actual learning, and will likely be tossed aside once the project completes. Whereas, developing a strong foundational knowledge in the topics I decided to attack, in a smaller scope such as just Rubber Ducky defence and the implications behind each Payload, incrementing complexity in a slow controlled manner led to the best possible results.

- **Hardware and Library Difficulties**
    - Accounting for development during shipping times and unsupported libraries
        - Prepare early and prepare hard. I ran into several difficulties such as shipping times (leading to dead time) and ordering hardware which had unsupported, extremely outdated libraries.

- **Unexpected bugs and Debugging PowerShell and gradual testing process development**
    - In the process of learning how to host a remote storage server and send local files to it using the SFTP (Secure File Transfer Protocol), entirely using PowerShell scripts. I faced bugs in every possible section of my code, from scripts to downloads causing a corrupt file download and authorization keys seemingly being incorrect. By failing to write my code in segments and instead opting to debug a bloated payload before even compiling, I had created a mess of bugs on top of bugs. To overcome this, instead of troubleshooting the code entirely, I cut the work down into sections, testing and debugging line by line until functionality was smoothed out. Ultimately, I learned that TDD (Test Driven Development) can apply even for Hardware applications and that the patience involved in TDD will save me time in the long run.

If I had more time

Given more time, I would have focused primarily on the deeper exploration of the application of hardware for penetration testing and how to defend against it. With my code mostly based around PowerShell scripting, I would explore methods outside of the PowerShell to attack a system. Furthermore, as I expand upon more complex attacks, I would like to investigate more complex exfiltration techniques such as what could be done locally with an added SD card onto the Raspberry Pi.

In addition, by adding the Dbisu library, I could access Ducky Script 1.0, a proprietary library used by an earlier version of the Hak5 Rubber Ducky which opens the doors to an entire library of payloads to compare and learn from.

Outside of the Rubber Ducky, I would want to explore further into hardware, creating a bare bones prototype of an RFID/NFC card spoofer and a Wi-Fi pineapple. The goal is to essentially build my own penetration testing kit and build a holistic understanding as to how these devices can be used in combination to test and improve robustness of a system defence.