<u>Hak5 - Rubber Ducky:</u>

The USB Keystroke Injection Tool

Team Members:

1. Chase Evans
2. Bryan Budusmith-Otoo
3. David Peñafiel

Professor: Casimer DeCusatis

Client: Melissa Chodziutko

Table of Contents:

Contributions:

Chase Evans:

- Cover page
- Table of Contents
- Group Resources
  - **Project GitHub**
  - **Final Presentation**
  - **All Resources**
- Group Project Plan
  - **All Sections**
- Hak5 Overview
- Hak5 Tools
- What is a RubberDucky
  - **All Sections**
- A Deeper Understanding of the RubberDucky
  - **All Sections**
- Advanced Scripting Techniques
  - **All Sections**
- Custom Script Library
  - **All Sections**
- Bugs and Problems Encountered
  - **All Sections**
- DuckyScript Syntax

Group Resources:

Official RubberDucky Documentation

Payload Scripting Site

Project GitHub

Group Project Plan

Final Presentation

Repositories/Code Examples:

- https://github.com/hak5/usbrubberducky-payloads/tree/master
- https://payloadhub.com/blogs/payloads/tagged/usb-rubber-ducky

Software Utilized:

- **Python 3.13.0** https://www.python.org/downloads/
- **PyInstaller (Compile Python scripts in a .exe file):**

  https://pypi.org/project/pyinstaller/
- **Visual Studio Code (IDE used for Scripting Python):**

  https://code.visualstudio.com

Articles:

1. https://blog.hartleybrody.com/rubber-ducky-guide/
2. https://null-byte.wonderhowto.com/how-to/load-use-keystroke-injection-payloads-usb-rubber-ducky-0176829/
3. https://hackmag.com/security/rubber-ducky/

Videos:

1. https://youtu.be/4DJDN2I456U?si=PsFI119GQmz75fgL
2. https://youtu.be/o1RbXtx0r4U?si=ZvCIo72lhSjtTpRQ
3. https://youtu.be/qCFgrUckmaE?si=o4I-FGPzMHGRz6Pw
4. https://youtu.be/h3ij6l3w-Eg?si=EjgwfS_SXXTT8jrC
5. https://youtu.be/_g16sQCapEE?si=vtd0Qu6YVc8nD_1s
6. https://youtu.be/QI-WrzOna3o?si=5_GgI3WUcOjg_dyw
7. https://youtu.be/1m8FYX6s-RU?si=uvZnwRDubfWbrZ9I
8. https://youtu.be/IxZcHTviKmA?si=5HeLLYbsF-CGLwyT
9. https://youtu.be/SMIvPE-37-0?si=VsO7JmRBM8KuRyNf
10. https://youtu.be/dJnMesIWL_4?si=84rx9rfjwgFoYdQA
11. https://youtu.be/bqNvkAfTvIc?si=IE_KcXRcxWx-9ni3

Books:

1. RubberDucky Guide for Hardware and DuckyScript

2. Hak5 USB RubberDucky Textbook (Physical Copy)

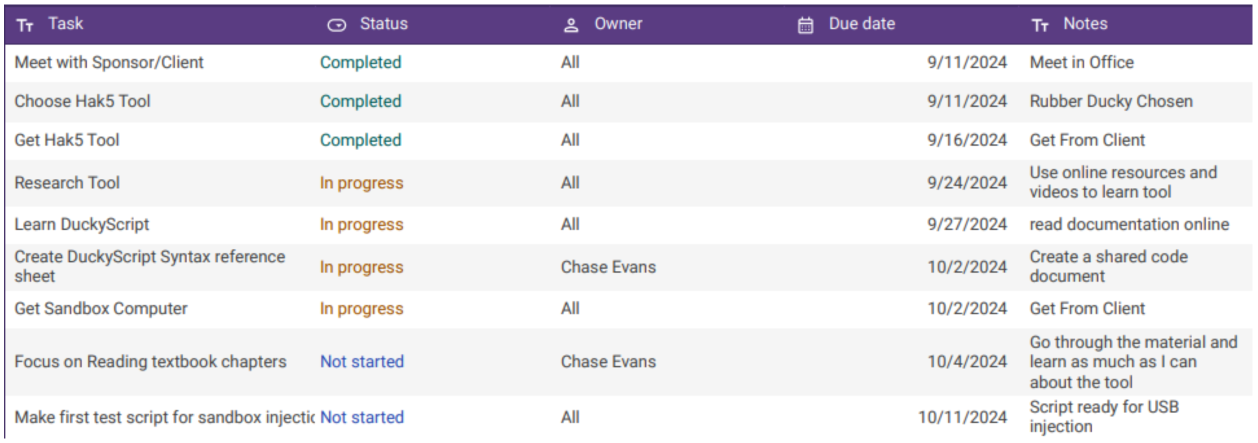
Forums:

1. https://forums.hak5.org/forum/56-classic-usb-rubber-ducky/
2. https://www.reddit.com/r/HowToHack/comments/u74jte/zip_bomb/

Group Project Plan:

Day one:

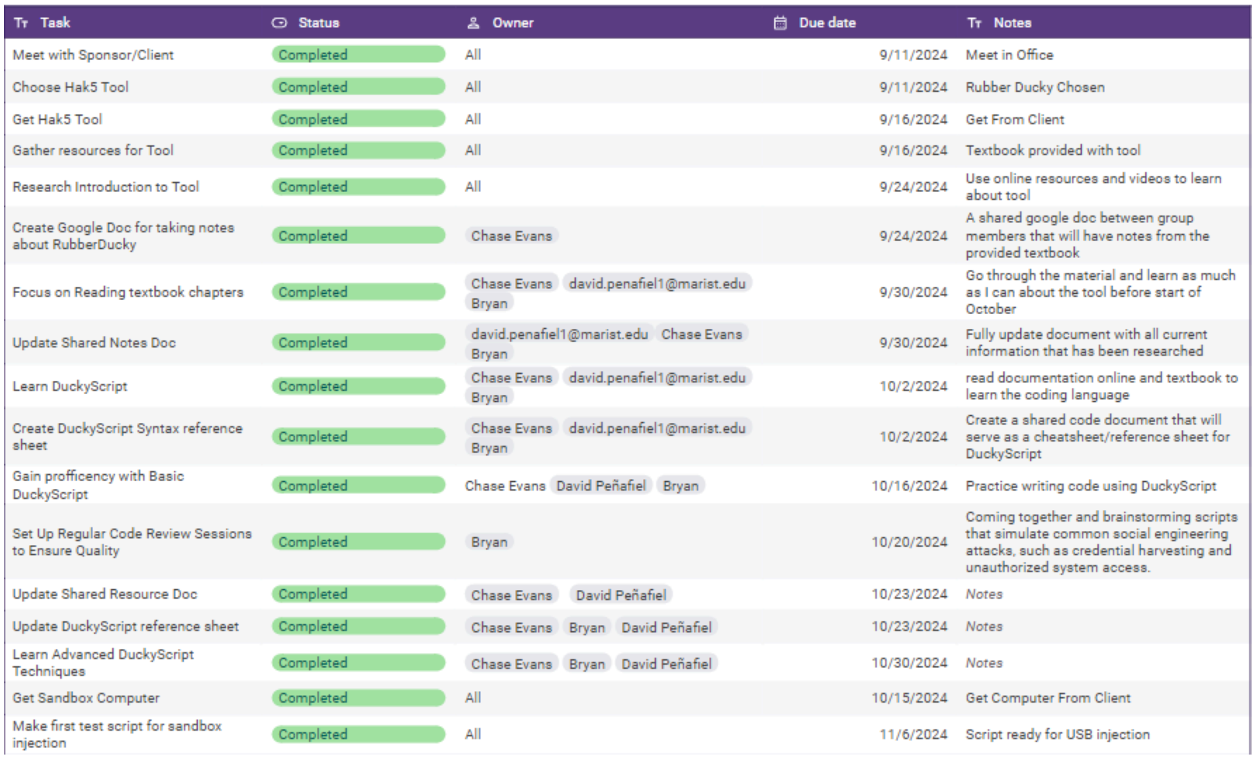
This was our original project plan on day one, we didn't really know what to expect from this project so we made sure that our plan was kept brief, and open ended. This was done to ensure we wouldn't lock ourselves into a path that we didn't fully understand yet. Overall, we wanted to get our footing with the tool, and how it worked before we dove into anything hyper specific.



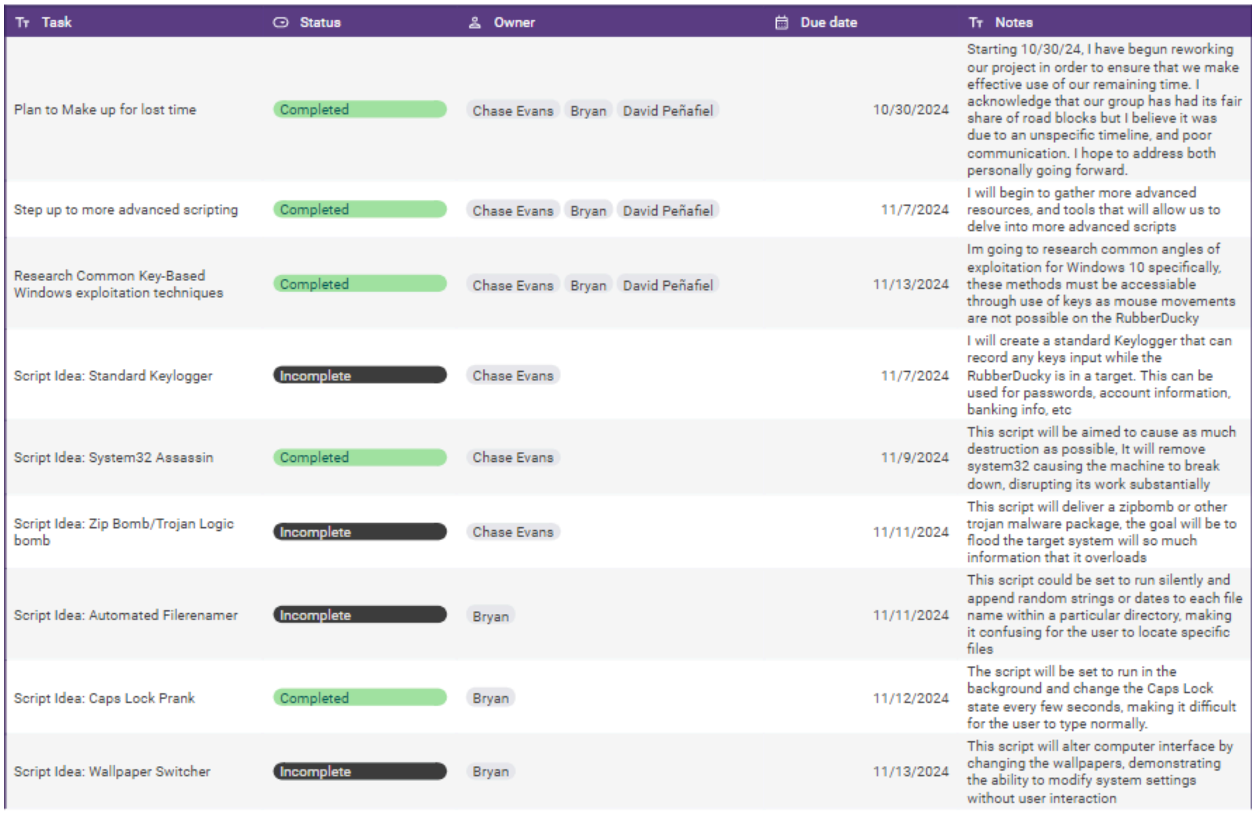
Task	Status	Owner	Due date	Notes
Meet with Sponsor/Client	Completed	All	9/11/2024	Meet in Office
Choose Hak5 Tool	Completed	All	9/11/2024	Rubber Ducky Chosen
Get Hak5 Tool	Completed	All	9/16/2024	Get From Client
Research Tool	In progress	All	9/24/2024	Use online resources and videos to learn tool
Learn DuckyScript	In progress	All	9/27/2024	read documentation online
Create DuckyScript Syntax reference sheet	In progress	Chase Evans	10/2/2024	Create a shared code document
Get Sandbox Computer	In progress	All	10/2/2024	Get From Client
Focus on Reading textbook chapters	Not started	Chase Evans	10/4/2024	Go through the material and learn as much as I can about the tool
Make first test script for sandbox injecti	Not started	All	10/11/2024	Script ready for USB injection

Final Version:

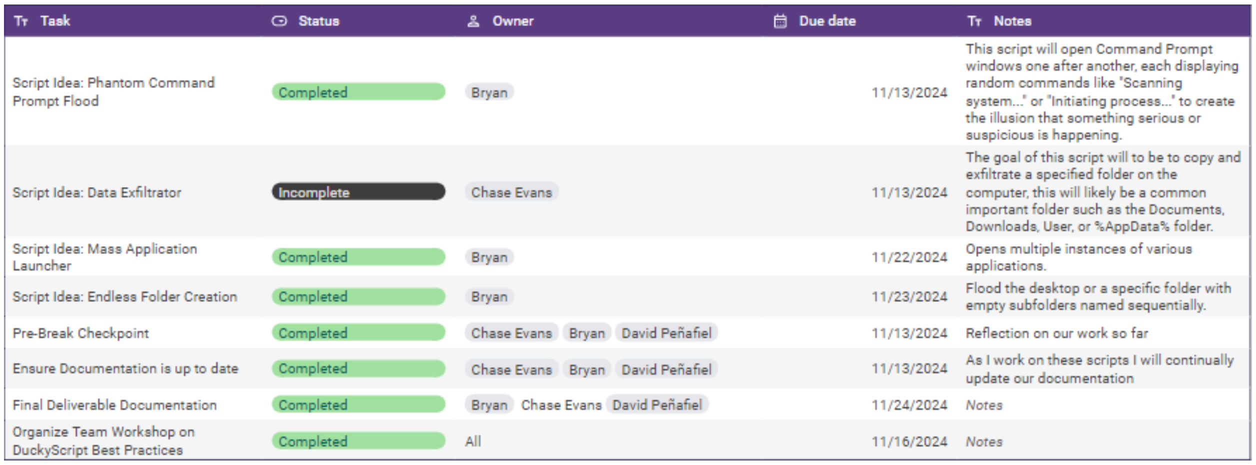
This is our final version of the project plan with the milestones we had accomplished. Overall we achieved most of what we wanted to by the end of the semester, of course there were some pain points as well. In hindsight, we should have begun technical development sooner as we spent a bulk of the first few weeks researching Hak5, their background, the RubberDucky itself, and what was possible. Some script ideas and areas of research also had to be dropped for the sake of time and prioritization of more crucial aspects of the project. For more information on the latter, please read our research to Create An Improvised RubberDucky.



Task	Status	Owner	Due date	Notes
Meet with Sponsor/Client	Completed	All	9/11/2024	Meet in Office
Choose Hak5 Tool	Completed	All	9/11/2024	Rubber Ducky Chosen
Get Hak5 Tool	Completed	All	9/16/2024	Get From Client
Gather resources for Tool	Completed	All	9/16/2024	Textbook provided with tool
Research Introduction to Tool	Completed	All	9/24/2024	Use online resources and videos to learn about tool
Create Google Doc for taking notes about RubberDucky	Completed	Chase Evans	9/24/2024	A shared google doc between group members that will have notes from the provided textbook
Focus on Reading textbook chapters	Completed	Chase Evans, david.penafiel1@marist.edu, Bryan	9/30/2024	Go through the material and learn as much as I can about the tool before start of October
Update Shared Notes Doc	Completed	david.penafiel1@marist.edu, Chase Evans, Bryan	9/30/2024	Fully update document with all current information that has been researched
Learn DuckyScript	Completed	Chase Evans, david.penafiel1@marist.edu, Bryan	10/2/2024	read documentation online and textbook to learn the coding language
Create DuckyScript Syntax reference sheet	Completed	Chase Evans, david.penafiel1@marist.edu, Bryan	10/2/2024	Create a shared code document that will serve as a cheatsheet/reference sheet for DuckyScript
Gain profficency with Basic DuckyScript	Completed	Chase Evans, David Peñafiel, Bryan	10/16/2024	Practice writing code using DuckyScript
Set Up Regular Code Review Sessions to Ensure Quality	Completed	Bryan	10/20/2024	Coming together and brainstorming scripts that simulate common social engineering attacks, such as credential harvesting and unauthorized system access.
Update Shared Resource Doc	Completed	Chase Evans, David Peñafiel	10/23/2024	Notes
Update DuckyScript reference sheet	Completed	Chase Evans, Bryan, David Peñafiel	10/23/2024	Notes
Learn Advanced DuckyScript Techniques	Completed	Chase Evans, Bryan, David Peñafiel	10/30/2024	Notes
Get Sandbox Computer	Completed	All	10/15/2024	Get Computer From Client
Make first test script for sandbox injection	Completed	All	11/6/2024	Script ready for USB injection



Task	Status	Owner	Due date	Notes
Plan to Make up for lost time	Completed	Chase Evans, Bryan, David Peñafiel	10/30/2024	Starting 10/30/24, I have begun reworking our project in order to ensure that we make effective use of our remaining time. I acknowledge that our group has had its fair share of road blocks but I believe it was due to an unspecific timeline, and poor communication. I hope to address both personally going forward.
Step up to more advanced scripting	Completed	Chase Evans, Bryan, David Peñafiel	11/7/2024	I will begin to gather more advanced resources, and tools that will allow us to delve into more advanced scripts
Research Common Key-Based Windows exploitation techniques	Completed	Chase Evans, Bryan, David Peñafiel	11/13/2024	Im going to research common angles of exploitation for Windows 10 specifically, these methods must be accessiable through use of keys as mouse movements are not possible on the RubberDucky
Script Idea: Standard Keylogger	Incomplete	Chase Evans	11/7/2024	I will create a standard Keylogger that can record any keys input while the RubberDucky is in a target. This can be used for passwords, account information, banking info, etc
Script Idea: System32 Assassin	Completed	Chase Evans	11/9/2024	This script will be aimed to cause as much destruction as possible, It will remove system32 causing the machine to break down, disrupting its work substantially
Script Idea: Zip Bomb/Trojan Logic bomb	Incomplete	Chase Evans	11/11/2024	This script will deliver a zipbomb or other trojan malware package, the goal will be to flood the target system will so much information that it overloads
Script Idea: Automated Filerenamer	Incomplete	Bryan	11/11/2024	This script could be set to run silently and append random strings or dates to each file name within a particular directory, making it confusing for the user to locate specific files
Script Idea: Caps Lock Prank	Completed	Bryan	11/12/2024	The script will be set to run in the background and change the Caps Lock state every few seconds, making it difficult for the user to type normally.
Script Idea: Wallpaper Switcher	Incomplete	Bryan	11/13/2024	This script will alter computer interface by changing the wallpapers, demonstrating the ability to modify system settings without user interaction



Task	Status	Owner	Due date	Notes
Script Idea: Phantom Command Prompt Flood	Completed	Bryan	11/13/2024	This script will open Command Prompt windows one after another, each displaying random commands like "Scanning system..." or "Initiating process..." to create the illusion that something serious or suspicious is happening.
Script Idea: Data Exfiltrator	Incomplete	Chase Evans	11/13/2024	The goal of this script will to be to copy and exfiltrate a specified folder on the computer, this will likely be a common important folder such as the Documents, Downloads, User, or %AppData% folder.
Script Idea: Mass Application Launcher	Completed	Bryan	11/22/2024	Opens multiple instances of various applications.
Script Idea: Endless Folder Creation	Completed	Bryan	11/23/2024	Flood the desktop or a specific folder with empty subfolders named sequentially.
Pre-Break Checkpoint	Completed	Chase Evans, Bryan, David Peñafiel	11/13/2024	Reflection on our work so far
Ensure Documentation is up to date	Completed	Chase Evans, Bryan, David Peñafiel	11/13/2024	As I work on these scripts I will continually update our documentation
Final Deliverable Documentation	Completed	Bryan, Chase Evans, David Peñafiel	11/24/2024	Notes
Organize Team Workshop on DuckyScript Best Practices	Completed	All	11/16/2024	Notes

What is Hak5?

Hak5 Overview:

Hak5 is the current industry lead for all things hacking. They have been creating award winning hacker tools for penetration testing, as well as immersive information security training since 2005.

Hak5 started as a podcast, the title of which also being Hak5. According to their website, it runs weekly and is interested in “covering everything from open source software and network infrastructure to penetration testing”. They also have several sister shows: HakTip, ThreatWire, Metasploit Minute and TekThing.

As a company they have released a large catalog of various tools, each serving a unique role in pen-testing as well as red-blue team competitions. They also provide immersive in-person training sessions to learn about information security, penetration testing, and how to effectively use their tools. These training sessions are heavily interactive and involve going up against “a host of hostile network adversaries” giving the people who attend the session invaluable practical experience.

Hak5’s Mission and Vision:

Hak5 over the years has been committed in both the making and evolving of cybersecurity tools to be made accessible to everyone. Their mission is to make sure that the role of ethical hackers is empowered with enough resources needed to help uncover and dissolve any vulnerabilities detected and thus create a safer digital environment. On the other hand, their vision is centered around bringing both technical expertise and practical solutions together to help enable individuals to protect systems and networks in this digital age. Their contributions to cybersecurity doesn’t go unnoticed as they are basically the cornerstone for the industry. They foster both collaboration and innovation as they help with insights to emerging trends and tools, how to uncover and tackle mishaps within systems, and overall provide the necessary techniques needed by cybersecurity professionals.

The Evolution of Hak5:

Since its commencement in 2005, Hak5 has grown from just being a niche podcast to quite frankly one of the biggest global leaders in its position centering around penetration tools and pentest training. It's key milestones was the release of the original Rubber Ducky in 2011, their launch of both their immersive and intuitive in-person training programs, and also lastly but not least, the ongoing of their successful weekly podcast shows. These key milestones help show their evolution and improvement over the years and thus also shows how it can ultimately adapt to cybersecurity threats. The Hak5 influence is not only about tools and training but stretches far beyond that. Their products are used in cybersecurity competitions, work/offices, academic settings and so on. An example would be how Marist college has integrated Hak5 tools such as the Rubber Ducky in efforts to provide information and knowledge about Hak5 to their students. This ultimately has resulted in Hak5 receiving abundant accolades for their impressive and innovative tools and society contributions. For instance, both the Rubber Ducky and WiFi Pineapple have become the standard tools for penetration testing in the industry.

Going forward, Hak5 is aimed towards its continuous expansion by offering some sort of insight or tools which align with emerging areas such as IoT security and most importantly, AI-driven threats. Advanced penetration tools will be needed for this and their commitment to innovation helps ensure that they will remain as the leader of the pack when it comes to cybersecurity operations, equipping both professionals and amateurs with the proper and right tools needed to tackle any cyber issue.

Hak5 Tools:

As stated above, there have been numerous tools produced and released by Hak5 for cybersecurity as a whole. Here are some of the tools Hak5 has to offer to the digital world:

- RubberDucky (Our tool of interest)
    - A USB keystroke injection tool that mimics a keyboard and thus delivers pre-programmed scripts rapidly.
- Wifi Pineapple

    - A device for wireless auditing, meaning it is capable of both identifying and exploiting vulnerabilities in WiFi networks.

- Key Croc
    - This is a covert keylogger which has been designed to record keystrokes and automate payloads.
- Bash Bunny
    - This is a versatile and diverse hacking tool which acts as a USB attack platform. It’s basically the perfect tool for red teamers.
- Wifi Coconut
    - This a tool designed to monitor and capture 2.4GHz WiFi packets across multiple channels
- Packet Squirrel
    - This is a network attack tool that intercepts and captures network traffic.
- Shark Jack
    - This is a portable device which is designed for quick deployment. It is done over the ethernet to extract both data and perform network recon.
- Screen Crab
    - This is more of a discrete device which captures and records HDMI output from screens.
- Plunder Bug
    - A small sized (like a bug) LAN tap for network monitoring.
- LAN Turtle
    - This is a covert backdoor for network penetration testing.

Hak5 Tool Diversification:

Below is a table showing the classification of each of these tools showing their right category:

CATEGORY	TOOLS	Its Primary Use
Keystroke Injection	Rubber Ducky and Key Croc	Automation of commands instantly once injected into the system
USB and Network Platforms	Bash Bunny and Packet squirrel	Multi functional and diverse range of testing capabilities
Wireless Auditing	WiFi Pineapple and WiFi Coconut	For WiFi security and traffic analysis
Physical Monitoring	Screen Crab, LAN Turtle, Shark Jack and Plunder bug	For stealthy observation and data collection from physical devices

Select Hak5 Tool Comparison:

The Rubber Ducky is a versatile keystroke injection tool but it's not the only tool in Hak5's catalog. We will be comparing our tool of choice (Rubber Ducky) to 2 others, the Bash Bunny and WiFi Pineapple. This will help better understand our tool's capabilities in reference to other tools.

Feature/Capability	Rubber Ducky	Bash Bunny	WiFi Pineapple
Main Purpose	keystroke injection for automated input attacks	Multi-purpose USB attack tool	Wireless auditing and network manipulation
Design	It seems as a regular USB drive	Also resembles a USB drive but slightly larger and bulkier	It is a small router-like device.
Attack Type	Keyboard emulation to inject malicious scripts	Versatile attacks including keystroke injection and file exfiltration	Network-based attacks on WiFi systems.
Users	This is typically for hardware based	General purpose penetration testers	Wireless security professionals.

	testers	(all round)	
Speed of execution	Rapid script execution through DuckyScript. (Extremely fast)	Moderate speed which is dependent on payload complexity.	Depends on WiFi traffic and scope. (Varies)
Ease of use	It uses simple scripting with Duckyscript.	This is more advanced as it supports multiple scripting languages.	Requires knowledge of WiFi protocols.
Power source	Powered directly by USB	Powered by USB	Powered by an internal battery
Compatibility	It's platform is windows.	Cross-platform (Windows, macOS, Linux)	Works with any WiFi-enabled device
Cost comparison	Lowest Cost $50 - $60	Moderate cost $100 - $120	Higher cost due to advanced WiFi $100 - $200

The WiFi Pineapple

The Bash Bunny

What is a RubberDucky

Overview:

A Rubber Ducky is a keystroke injection tool disguised as an ordinary USB flash drive. It emulates a keyboard, allowing it to inject pre-programmed scripts into a target system at rapid speeds. These scripts, written in DuckyScript, can automate complex tasks, exploit vulnerabilities, or execute commands covertly. The Rubber Ducky is a powerful tool for testing system security against unauthorized access and malicious payloads.

There have been two iterations of the RubberDucky as well as it’s native scripting language, DuckyScript

The RubberDucky Mark I (pictured below) came out in 2011 alongside DuckyScript 1.0:

The latest iteration of the RubberDucky has been the Mark II (pictured below), coming out in 2022 alongside the launch of DuckyScript 3.0:

For this project we worked with the Mark II and DuckyScript 3.0 exclusively, so all information within this documentation applies to that model and language.

How it Works:

The RubberDucky operates using its onboard CPU to deliver keystrokes at a speed far exceeding human capability. It exploits the inherent trust that computers have in human input. The operating system treats it as a trusted human interface device, no different from a keyboard. Its versatility lies in the flexibility of DuckyScript, which enables users to write tailored payloads for specific tasks, from privilege escalation to data exfiltration. The device's plug-and-play nature and double USB system enables compatibility across Windows, macOS, and Linux systems.

Software:

- Payload Studio
    - https://payloadstudio.hak5.org/community/
    - An Integrated Development Environment consisting of:
        - Source code editor
        - Compiler
        - Encoder
        - Debugger
- Inject.bin

    - The binary equivalent of the DuckyScript source code generated by the compiler and encoder
    - Must be placed in the RubberDucky root directory in order to launch when plugged into a device

- Internal Disk Layout:
    - Docs - This folder is used to store links to external resources
    - Payloads - this folder is used to store payloads and scripts
    - Hangar_bay - this is a custom folder created for our RubberDucky to house python scripts that will be injected

Hardware:

- CPU with onboard flash
- High Speed USB 2.0 Interface
- Micro SD card reader
- Micro push button
- Multi color LED indicator
- Standard USB type A and C connectors

Payloads:

Rubber Ducky payloads are scripts written in DuckyScript, designed to execute automated tasks or exploit vulnerabilities on target systems. These payloads can range from simple commands, like opening a website or downloading files, to more complex actions, such as privilege escalation or system reconfiguration. Payloads can include keystroke loggers, reverse shells, and data exfiltration tools, allowing penetration testers to simulate real-world attacks. Advanced payloads often utilize multi-stage processes, combining rapid injection with external scripts hosted on servers to extend their capabilities.

Internal Storage Folder Layout:

<<<NOTE>>>

Each RubberDucky Layout may differ from device to device based on how the user decides to set it up.

For the sake of our documentation this is how we had ours organized and we recommend creating a structure similar for proper organization

| DUCKY (D:) - Root Directory of the RubberDucky

| — docs - folder containing important and useful links/resources by Hak5

| — hangar_bay - custom folder created by our group to store Python scripts

| — payloads - folder containing subfolders for each stored payload

| — — <payload> - folder named after payload, contains a payload.txt and inject.bin

| — — — payload.txt - source code of the script (Keep for reference)

| — — — inject.bin - the compiled source code (The actual script)

IT IS IMPORTANT TO NEVER RENAME THE <hangar_bay> FOLDER

THIS WILL BREAK SCRIPTS THAT REQUIRE A SPECIFIC FILE PATH

User Setup Guide:

If you are using a Brand New RubberDucky, following the instructions below

If you have a RubberDucky the has been set up previously, skip to the Use Guide

This setup guide below will explain how to set up the internal button for the USB RubberDucky

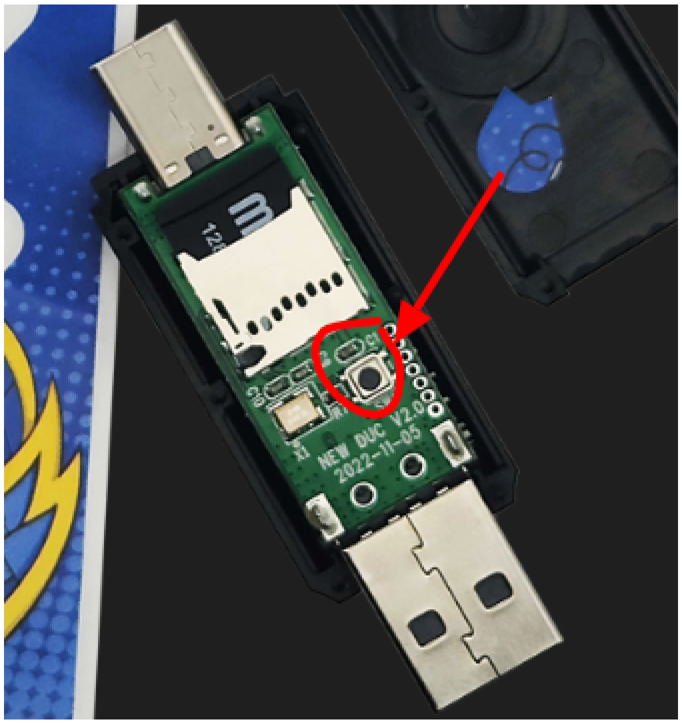
1. Take RubberDucky out of its packaging, you should have 3 items
    a. RubberDucky device
    b. Sticker Pack
    c. User pamphlet with resources and help guide

2. Place all items aside and grab just the RubberDucky

3. Carefully remove the USB-C end cap and the metal hinge

4. Alongside the RubberDucky there is a seem, use your nails, tweezers are other precision tool to carefully open the case

5. Within the case look for the small gray button on the motherboard, reference is pictured below:

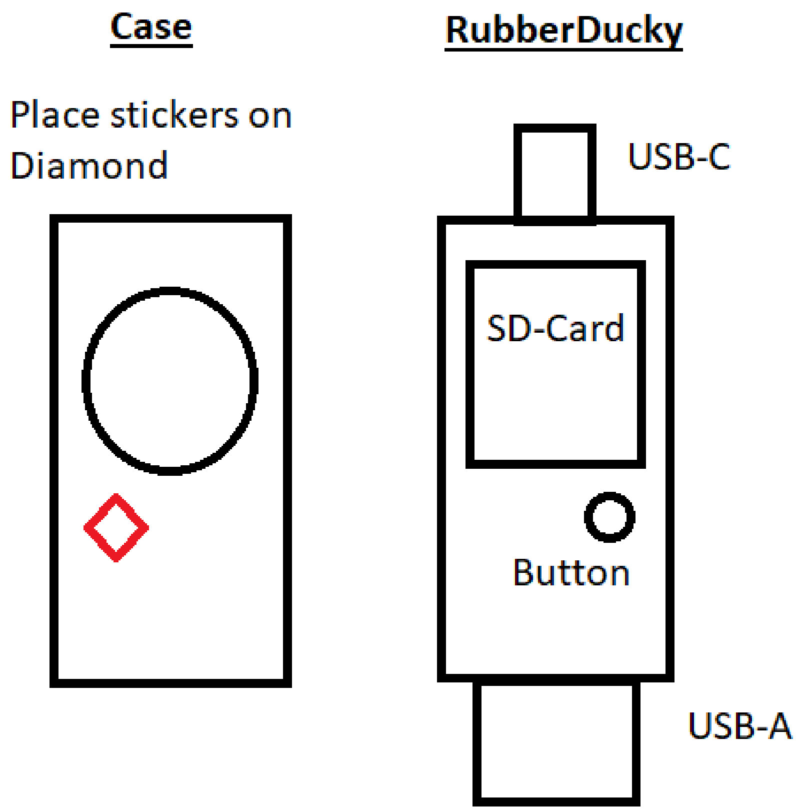


6. Grab the sticker sheet from before, on the right side there are 4 teardrop shaped stickers (They are blue stickers on a blue background so they may be hard to see)

7. Take some of these stickers, we recommend 2 - 3, and layer them on top of each other to create a single, thick, sticker

8. Position the RubberDucky so the USB-C (the thinner USB) to be on top, and the larger one to be on the bottom

9. The internal button should be on the lower-middle right of the RubberDucky, you placed the stickers on the case mirrored on the lower-middle left. This way when the RubberDucky is put back together, the stickers will be on top of the button (Below is a diagram of where to place it)



10. Place the layered stickers on the Red Diamond on the diagram

11. Close the case back around the RubberDucky and squeeze it by the USB-A

12. Listen for a distinct but faint *CLICK*
    a. If you hear this, congrats! The RubberDucky is officially ready to go
    b. If not, you will have to reopen the case and add another sticker OR you may have misplaced the sticker, check and try again

<Continue to Device Use Guide Below>

Device Use Guide:

Accessing the RubberDucky's Storage

1. Plug in RubberDucky to the Device
    a. If you are unsure if the RubberDucky is armed or not, we recommend using a disposable laptop or computer
    b. This will be referred to as the <sandbox> computer

2. Open your File Explorer App

3. Scroll down to the bottom on the left side containing your quick access folders and system folders

4. At the bottom beneath your OS (C:) drive should be another drive called DUCKY (D:)

5. If this is not visible, click the internal button within the RubberDucky to switch it into Storage mode
    a. **If you have not set up the internal button, please see the *<User Setup>* Guide above**

6. You should now be able to see the DUCKY (D:) drive

7. Click on the drive to enter the RubberDucky's storage

Setting up Payloads:

For this section please read the above ***<Internal Storage Folder Layout>*** Section to get an understanding of how the internal layout is.

- **How to disarm a RubberDucky / Check if it’s armed:**
    - Open the devices Root directory <DUCKY (D:)>
    - If you see an inject.bin file in this folder, the device is currently armed with a payload
    - Delete, Rename or Move it out of this folder to disarm it
        - We recommend Deleting the file to avoid duplicate scripts
        - All scripts should have a backup stored within their assigned payload folder that will be simply copy and pasted from. This ensures I true script per payload

- **To arm the device with a payload:**
    - Open the payload folder within the Root directory
    - Find the chosen payloads folder
    - Open it and copy the inject.bin file
        - ***NEVER DELETE THIS FILE UNLESS YOU WISH TO REMOVE THE SCRIPT ENTIRELY***
        - There is no way to recover deleted files on the RubberDucky itself
    - Paste this inject.bin file into the Root directory
    - If you get a popup for an existing file, press replace all
    - You have now armed the RubberDucky
    - Unplug it and plug it into the target device to automatically activate the payload

<<<NOTE>>>

If you rename the inject.bin file to anything else, it will not launch the script upon connection, the name MUST stay the same.

A Deeper Understanding of the RubberDucky:

How the RubberDucky Leverages Social Engineering:

The biggest danger posed by a RubberDucky is not necessarily the payload, but it’s inconspicuous appearance. Without any identifying markers, it looks like any other flash drive. If left around a workspace, anyone curious enough could plug it in and cause immense damage almost immediately. Some scripts are entirely silent and hidden so they could connect it to a system without ever realizing what they have done.

Inherent Trust Between Computer and Human:

Computers, by design, trust input from humans with absolute certainty in your actions. A computer will never second guess the instructions it is given via a keyboard or mouse. The RubberDucky exploits this Inherent Trust by disguising itself as a Human Interface Device (HID). This means any command given to a target system by the RubberDucky will be given the green light. The attacker must only concern themselves with the restrictions set on the system by another human.

Ethical Considerations:

When doing anything with the RubberDucky you must always keep in mind that this is a powerful penetration testing tool, and that power can cause severe damage if handled irresponsibly or recklessly. This makes it a vital resource for ethical hackers as they approach situations with care and responsibility. Ethical considerations are needed to ensure that this tool is used in a manner that protects clients and also complies with both the laws and upholds integrity.

Here are some ways you to ensure you are using the RubberDucky ethically and safely:

- Lay out everything to your client clearly

- Clearly explain what tools and techniques will be used, including Rubber Ducky payloads.
- What damage they are capable of causing
- If there are any lingering effects
- Try not testing outside the agreed space of testing to help avoid any legal and ethical implications.

● Ensure Data Privacy

- Do not access or change any personal data/information unnecessarily during testing.
- Make sure you are anonymous or use test systems not personal systems to minimize risks.
- If testing involves exposure to sensitive information, you have to make sure to protect and delete it right after testing.

● Educate The Client

- Make sure your client understands the purpose of the testing and why the results matter
- Make sure you provide insights into the imitations of tools like the Rubber Ducky, which highlights that no tool full proof guarantees security.

- Make them aware of training available so they may be aware of recognizing and avoiding malicious USB devices.

● Communicate Regularly with Stakeholders

- Make sure you provide updates with your client throughout the entire process.
- Make sure to maintain transparency, especially when unexpected issues arise during testing.

● Ensure all parties have given proper consent to the pen-test

- Always make sure you have proper and written consent from the client before any type of penetration testing, this extends outside of this tool as well.
- Make sure to avoid “gray box” testing unless you are in the green to do so as it may lead to unwanted damages.
- The authorization of testing should outline the systems and networks that will be used for the pen test.

● Follow Client’s/Organizations rules and policies

- Make sure testing aligns with the set rules the client proposed.
- Make sure to try avoiding violating any company rules or trying to bypass any regulations unless instructed to do so.

- Mitigate potential damages

- Make sure to test payloads in a controlled environment (SandBox) before you deploy in the client's system
- Make sure to include some form of safeguards in the scripts such as delays to help prevent an accidental execution on unauthorized systems.

Defending Against a RubberDucky:

Defending against a RubberDucky attack can be tricky due to its unassuming appearance. One can be plugged into a system and not get so much as a double take. It's physical nature also makes it difficult for antivirus **software** to detect, which helps emphasize physical security and user awareness are critical in this case of defense strategy.

The best precautions you can take are:

- Ensure only approved devices are allowed to interact with important systems
- Inform employees and security to keep an eye out for unattended flash drives.
- Ensure that critical infrastructure or systems are kept secure behind security checkpoints and/or doors that can lock, ideally via keycard
- Install software that can block connections from unauthorized or unknown devices.

- Conduct regular awareness sessions to help highlight real-world examples of malicious USB.
- Install monitoring software that logs all USB activities.
- Conduct regular pentesting sessions using tools like the Rubber Ducky to help evaluate and improve defenses
- You can also test defenses by also conducting random employee awareness tests simulating USB attacks.
- Use firewalls and intrusion detection systems (IDS) to both detect and block unauthorized data exfiltration.
- Restriction of USB ports on critical systems.
- Using encrypted USB devices that require authentication.
- Solutions that can disable USB ports if devices are used outside the approved locations (Geofencing USB devices).
- Adopting the Zero trust framework which basically means assuming all external devices are untrustworthy unless it has passed through some form of rigid verification process.
- Most importantly, do not interact with devices that do not belong to you, or you don’t know the contents of.

Limitations of The RubberDucky:

We can all see the Rubber Ducky is a very powerful tool when it comes in contact with its desirable host for penetration testing and training as a whole. Alas,the tool being so almighty, it has its limits, and these limits should be carefully considered as it can heavily impact your progress with the tool or not.

- Dependency on physical access:

- The Rubber Ducky needs a physical device as its host to showcase its abilities. Without this physical access it is rendered useless. This limits its capabilities as it solely requires access to a USB port to execute payloads.

- Lack of Payload Complexity:

- DuckyScript is a versatile language. It is not as complex or as advanced as the other scripting languages which makes it easier to deploy. During this project for example, just DuckyScript was not enough to sample the more complex automations as other languages such as python needed to be integrated for the more advanced results. This limits its range of capabilities as it needs to be amplified by other scripting languages to perform higher and more complex tasks.

- Increased awareness and training

- One thing that also limits the RubberDucky is the increased awareness and training. Corporations have now integrated policies for employees to undergo training to enhance both awareness and knowledge to be able to detect devices (such as the RubberDucky) which may be harmful to critical systems thus limiting the full potential of the RubberDucky.

- Lack of Encryption of Payloads:

- One main liability of the Rubber Ducky is the lack of already built-in encryption for its scripts/payloads. The codes are stored in plain text on the device's microSDcard. This makes the device information easily accessible to anyone with sufficient knowledge and can easily read and analyze the scripts present.

Overall, the Rubber Ducky is a very powerful tool for both education and penetration testing, however these limitations make it less efficient in much more secure environments.

Common Use Cases:

The Rubber Ducky is beneficial in multiple ways as these ways are shown in use cases. We have categorized these use cases into four subheaders Technical, Training and Awareness, Research and Development, and lastly Real-world scenarios.

Technical Use Cases

- By simulating unauthorized access to systems, the injection of the Rubber Ducky is done to automate its commands to exploit vulnerabilities.

- Used in privilege escalation which in this case means testing how well systems restrict access to unauthorized devices by trying to gain permissions.
- It is also used in payload automation as it is deployed to run preprogrammed scripts to produce repetitive tasks like network scans and password resets.

Training and Awareness

- Employee security training is valid here. They must be exposed to the risks of plugging in unauthorized devices into vulnerable machines. There should be inclusion of exercises to help employees identify and report any device seemingly suspicious.

- There are awareness campaigns where tools like the Rubber Ducky are showcased to all during corporate security workshops.

Research and Development

- In this case, testing new exploits i.e. experimenting with new attack vectors in a controlled environment to witness vulnerabilities and ways to combat or amplify those vulnerabilities.

- There are often studies on how to improve tools and thus tool development is a thing. After research and studies, one must create or refine payloads for specific cases such as bypassing MFA.

Real World Scenarios

- Social engineering campaigns are performed as dropping USB devices in public areas such as libraries or cafe’s which helps run a statistic of how often people will randomly pick it up and plug it in their system.

- Also, testing how resilient the network of enterprises fair against hardware tools. This helps identify any gaps in specifically USB port control policies.

<u>Advanced Scripting Techniques:</u>

Integrating Python scripts into payloads:

<u>Dependencies:</u>

- **Python** - Current Version Encouraged
- **Pyinstaller** - Used to create .exe files
- **IDE Compatible with Python for Scripting**
    - We recommend **Visual Studio Code**

Utilizing a RubberDucky to inject Python scripts into a target system is a very powerful way to exploit vulnerabilities. A python script compiled into an .exe file can linger on a system, long after the RubberDucky has been removed. It may also be set as a startup program to ensure it will always run. The main shortcoming of this method is the increased delivery time, averaging 30 seconds to 1 minute depending on the script size and target system speed. Another thing to keep in mind is that these scripts are likely to be blocked by antivirus, requiring privilege escalation or encryption to function undetected.

This is only one way to incorporate other scripting languages into a RubberDucky Payload. Our group focused exclusively on python but theoretically, any scripting language can be injected onto a system through a payload. This is true as long as that language supports a method, officially supported or third party, to compile code into pre-packaged .exe files, or similarly compact single file programs. This method ensures size is kept to a minimum to accommodate the RubberDucky's small storage size and injection speed for large files. This also ensures that the target system doesn't need to be preinstalled with any software to make it run the script. This drastically increases the payload's potential target count.

Creating a Homemade Improvised RubberDucky:

Unfortunately this process was unsuccessful, as lots of roadblocks were hit and it had to be put on the backburner in favor of more crucial aspects of the project

Below is a summary of the research and what we managed to achieve:

For this project, Chase Evans conducted research into the possibility of creating our own improvised RubberDucky's utilizing just an ordinary, everyday USB flash-drive and some python scripts. This would, in general, have less features than the actual RubberDucky by Hak5 as it would lack most of the hardware that makes the tool special. However, if successful this would open up a lot of new opportunities for how a RubberDucky adjacent tool can be used.

The concept held in mind was the idea of bulk buying a ton of cheap flash drives online, converting them all into improvised RubberDuckies using python scripts and a program to inject them into the target. After setting them up it would be possible to leave them around public spaces like train stations, airports, or office buildings. Once planted, they would wait for someone curious enough to take the device and plug it into their system. This would lead to them unknowingly or unexpectedly launching the payload onto their system. This payload can be anything but some top picks could be:

- Trojan virus
- Ransomware software
- Install a backdoor
- Data exfiltrator (Send their personal info over the internet to a private server)
- Zip Bomb
- Delete their System32 and crash the system

Essentially anything could happen, it would just take some additional research for each payload, no different from scripting with DuckyScript.

Steps needed for this to be a reality:

- Standard USB Flash drive to house the payload
- Python script that will be injected into the target

- Software to turn Python scripts and their modules into a prepackaged .exe file
- Software to allow the script to be run automatically when the device connects

What we had:

- 2 Standard USB flash drives
- A test python script that would open a new .txt doc and type “Test complete”
- Pyinstaller: an extension mentioned previously in the doc that compiles python code and modules into 1 bundled .exe
- USB AutoRun Creator software

All of the pieces were there expect for unfortunately, the software we needed to run the script, arguably the most important piece. The original plan was to utilize USB AutoRun Creator by SamLogic software. This software allows a user to set specific .exe files stored on a flash drive to autorun when a device is connected. If we set the python .exe created with pyinstaller to utilize this program we would have our improvised RubberDucky concept complete and we could move forward to optimizing it further. This is where the complications came into play.

1. USB AutoRun Creator does not work well with most SSDs as it was made when HDDs were still the vast majority of storage devices. This is a problem because today many people have SSDs as their primary storage type

2. USB AutoRun Creator requires the software to already be installed on the system for the autorun to function consistently. This is a major obstacle severely limiting the capabilities of this method
3. The final nail in the coffin was found during the later day's of research. Windows 10 has put systems in place to block USB storage devices from autorunning programming upon connection, specifically for reasons like this mini-project. This means this method could only work on Windows 7 and prior systems. Again this severely limits this method.

The reason the RubberDucky is able to get around all of these roadblocks is due to it’s special hardware. It’s onboard cpu and processor take over the role of USB AutoRun Creator all within a smaller and easier to use package. It is also to get around Windows blocking autorunning USB devices by tricking the computer to read it as Human Interface Device (HID) instead.

Concluding this research, as a group we have come to appreciate the technical marvel that is the RubberDucky even more. Trying to replicate even a fraction of its capabilities was quite the task. Given more time we may have found a similar work around however, given our limited timeframe this was not an ideal use of our project’s time and resources.

Custom Script Library:

Chase's Scripts

The PhantomCrasher

Dependencies:

- PayloadStudio (DuckyScript)
- Windows Powershell
- Administrator access on the target system
- Python
    - Module Dependencies:
        - Random
        - Time
        - keyboard
- Pyinstaller (to compile python script into a single .exe)

Method of Delivery:

1. Plug in RubberDucky to target system
2. Wait for device to confirm connection
3. Press internal button on RubberDucky to inject the payload
4. Wait for the Scripts completion
5. Remove the device from the target system

Description:

This script was designed to inject a python script into the target systems Startup folder. It does this by obtaining admin access to the Windows Powershell and copying the script from the RubberDucky to the target folder. The file is also named in a manner that mimics a legit windows process, meaning even if found the victim will likely be too nervous to actually remove the file. Once this has completed, the script disables Window Defender’s ability to scan for it, then the script runs silently in the background. The script is run in the background processes with no trace on the desktop or taskbar meaning it is entirely noticeable unless you know exactly where to look. While running in the background, this script has a countdown that infinitely loops at random durations. Each time the countdown finishes, it crashes the currently opened application without sending any feedback, or pop-up. This is extremely disruptive as there is no way to stop it without knowing the exact file path that it was placed into. For the victim it will be impossible for them to figure out why all of their apps keep crashing, meanwhile the script is in the background the whole time, like a phantom.

Strengths:

Due to its hidden nature while running, lack of console outputs, and ability to run on startup when a computer is turned off and on, this script has the potential to persist on a system indefinitely. If used against a victim who is not tech savvy this script is practically invisible and impossible to find. It would require someone with very advanced knowledge of the Windows Operating system to find the process amongst the default windows processes, kill it, and then scour the files to remove it before it has the chance to restart.

Shortcomings:

The main shortcoming of the payload is that it requires the target computer to have admin access on their account. This means this script will only work on personal, single user devices, or with managers/admins of companies. You will not be able to use this script on any desktop found in an office building or similar environment. The other shortcoming is that this script can only account for Windows Defender protected computers. If the target has other antivirus software installed, it is likely to be detected and its runtime prevented.

Haunted.exe

Dependencies:

- PayloadStudio (DuckyScript)
- Windows Powershell
- Administrator access on the target system
- Python
    - Module Dependencies:

        - Random
        - Time
        - keyboard
- Pyinstaller (to compile python script into a single .exe)

Method of Delivery:

1. Plug in RubberDucky to target system
2. Wait for device to confirm connection
3. Press internal button on RubberDucky to inject the payload
4. Wait for the Scripts completion
5. Remove the device from the target system

Description:

This script works using the same techniques utilized by the PhantomCrasher payload. It plays a timer in the background of the system, this time far longer than the PhantomCrasher, going off every handful of hours. When the timer hits zero the computer will speak to the user and deliver a creepy message with the goal to scare the user as they aren't expecting their computer to speak.

Strengths:

This script is very good at it's goal of surprising and scaring the target user. This is the only real application of the payload as it doesn't tamper with anything on the computer. I would argue that the presence of a voice coming from your computer would cause a person to freak out and feel a bit crazy. Desperately searching through their computer trying to find the source or even bringing the device to a forensics team, wasting valuable time and energy.

Shortcomings:

The main shortcoming of the payload is that it requires the target computer to have admin access on their account. This means this script will only work on personal, single user devices, or with managers/admins of companies. You will not be able to use this script on any desktop found in an office building or similar environment. The other shortcoming is that this script can only account for Windows Defender protected computers. If the target has other antivirus software installed, it is likely to be detected and its runtime prevented.

System32 Assassin:

Dependencies:

- PayloadStudio (DuckyScript)
- Windows Command Line
- Administrator access on the target system

Method of Delivery:

1. Plug in RubberDucky to target system
2. Script will run automatically, displaying a warning message explaining what the following script will perform
3. Press the internal button to confirm script delivery
4. Watch as the computer breaks

Description:

This script was designed to cause as much damage to a target system as possible. What better way to destroy a system than to remove all of its core functions and files. This script when run will first create a txt file and type out a warning explaining the scripts function and the severity of what it can do. If the attacker wishes to continue they press the button, launching the rest of the script. This will cause the windows command line to open in administrator mode, enter the System32 directory, and remove it from the system. Once removed there is nothing you can do but watch the computer slow down, freeze, and break entirely as all of its core system dependencies are stripped away.

Strengths:

The strength of this script is that it has the capabilities to destroy a target system using a single line of code within the command line. If pure destruction is the intent against a device, there is no simpler and more direct way to do it than this script.

Shortcomings:

The main shortcoming of the payload is that it requires the target computer to have admin access on their account. This means this script will only work on personal, single user devices, or with managers/admins of companies. You will not be able to use this script on any desktop found in an office building or similar environment. The other shortcoming was added intentionally for the sake of testing in a safe environment. The warning message that pops up serves as a way to make the execution process deliberate, avoiding any accidental executions during testing. In a final, fully tested and live version of this script, the warning would be skipped and the payload would simply delete System32 as soon as the device is connected.

WindowWiper:

Dependencies:

- PayloadStudio (DuckyScript)

Method of Delivery:

1. Plug in RubberDucky to target system

Description:

This script, when injected into a system, will instantly start an infinite loop between 2 keys, Alt F4 and Alt Tab. The combination of these keys will cause the computer to close and switch between every open application on the target's desktop. The keystrokes happen so fast that it will seem like every app has crashed simultaneously and the mouse and keyboard will be locked during the entire process. This script's usefulness is very situational, it was designed with the intention to disrupt important processes that are ongoing on the target system. This can include canceling downloads, disconnecting from important services, or in more destructive cases, shutting down an application running a critical service.

Strengths:

This is a very simple script and requires no dependencies outside of DuckyScript, which all payloads already utilize. Due to its simplicity, the failure rate is 0, this script will always run and does not require any administrative permissions.

Shortcomings:

This script only runs when the RubberDucky is connected with the device, once removed the loop will break. This is not a problem if the intention was to quickly wipe the desktop but if any persistent threat was expected it will not meet expectations.

Whats Up Doc?:

Dependencies:

- PayloadStudio (DuckyScript)

Method of Delivery:

1. Plug in RubberDucky to target system

Description:

This is a very basic script, it was the first I created to get the hang of DuckyScript. It is also great at showing off the Keystroke injection speeds capable by the RubberDucky, I used it as a demo in our final presentation. All this script does is create a new text document and rapidly type out ASCII art of Bugs Bunny saying his iconic line.

TheClassicBlunder:

Dependencies:

- PayloadStudio (DuckyScript)

Method of Delivery:

1. Plug in RubberDucky to target system

Description:

This is another basic payload like What's up Doc? This script was made to test out how fast the RubberDucky could inject and run a file onto a target system. When plugged in, the script will open the command line, inject a file from the hangar_bay, and play it. The file in question is a .mp4 file with the infamous song Never Gonna to Give You Up by Rick Astley

Forced Telnet Connector:

Dependencies:

- PayloadStudio (DuckyScript)
- Administrator access on the target system

Method of Delivery:

1. Plug in RubberDucky to target system

Description:

When this script is run, it will open the windows command line in administrator mode, enable telnet, and then attempt to connect to any given address. This will be set prior to injection, allowing any possible IP or server to be the target's forced connection.

Strengths:

This script is only limited by the hackers imagination and technical know-how about networks. The connection made can be used for nearly anything from remote access, backdoor injection, network credential harvesting, etc. The computer is entirely up to the mercy of whatever connection it has established. This can be devastating for the target system and lead to untold damages.

Shortcomings:

This script is untested, due to Marist's strict policies about unauthorized network connections our team was unable to provide any tangible evidence of this script's devastating effects. That being said, this script will do its job of forming a connection to any assigned address if the target system has administrator access and a non-managed network. This script is most effective on personal systems on public and/or personal networks.

Bryan's Scripts

MASS APPLICATION LAUNCHER

Dependencies:

- Payload Studio (DuckyScript)
- Access to Target System
- Windows OS with default apps such as Notepad, Calculator and Paint.

Method Of Delivery:

- Plug Rubber Ducky into the target system
- Once the device confirms the connection, press its internal button
- This initiates the payload
- Remove the Rubber Ducky once the operation has been achieved. (It is a rapid event)

Description:

This script was designed to overwhelm the user by creating multiple instances of default applications already present on the target system such as notepad, calculator, and paint. Once activated, the payload delivers the DuckyScript to execute commands at crazy speeds, repeatedly launching the applications. These applications cause disruption in the workflow of the user. This script's simplicity is its strength as it requires no additional dependencies, only the default applications will suffice.

The script made:

```
REM TITLE: Mass Application Launcher
REM AUTHOR: Bryan
REM DESCRIPTION: Opens multiple instances of various applications endlessly.

DELAY 1000
REPEAT
   GUI r
   DELAY 500
   STRING notepad
   ENTER
   DELAY 200
   GUI r
   DELAY 500
   STRING calc
   ENTER
```

```
DELAY 200

GUI r

DELAY 500

STRING mspaint

ENTER

DELAY 200
```

Strengths:

- This script was straightforward, making use of the already present apps. The ease of use was guaranteed.
- In addition to it being easy to use it was harmless as well, causing no type of long or short-term damage.

Shortcomings:

- It is easily detectable. The user will right about notice what is happening and can quickly terminate it by rebooting.
- It has no aftermath effects as disruption will stop immediately once script has been terminated and thus no lasting impact on the system.

ENDLESS MESSAGE BOX FLOOD:

Dependencies:

- Payload studio (DuckyScript)
- Windows Command Prompt
- Access to target system

Method of Delivery:

- Plug in RubberDucky into target machine
- Activate the payload through the Ducky's internal button

Description:

This script was designed with a simple VBScript through command prompt to display an infinite amount of message boxes. The message boxes contained customizable text to further confuse the user. This was done in an infinite loop which creates an environment not productive at all for the user, as it forces them to reboot the entire system.

The Script made:

```
REM TITLE: Endless Message Box Flood
REM AUTHOR: Bryan
REM DESCRIPTION: Displays continuous pop-up messages endlessly.

DELAY 1000
GUI r
DELAY 500
STRING cmd
```

```
ENTER

DELAY 1000

STRING :loop

ENTER

STRING echo x=msgbox("Your computer is running low on resources!",0+16,"Warning")

ENTER

STRING goto loop

ENTER
```

Strengths:

- It is highly and I repeat **highly** annoying. These endless pop-ups create both intrusive and disruptive events and thus making the system unusable.
- Another strength is that the messages are customizable, meaning you can change it and make it say whatever you want it to.
- It is a low resource usage script, meaning it requires very minimal system resources to function hence making it difficult to detect through performance monitoring.

Shortcomings:

- There is no stealth factor to this script as the visible nature of the pop ups instantly give it away thus making it easy to identify and halt the attack.
- It requires a command prompt, meaning, if the command prompt is disabled or restricted for whatever the reason, it will not work.

AUTOMATED FILE RENAMER:

Dependencies:

- PayloadStudio (DuckyScript)
- Access to target System
- Windows Command Prompt
- Target Folders with Files

Method of Delivery:

- Insert Rubber Ducky into target system
- Activate payload to execute the script through the command prompt

Description:

This script was designed to create confusion and frustration to users as files would be renamed in target folders. It adds _random to each file name. For example if your file was saved as Capping1.txt it becomes Capping1.txt_random. It further recycles it and makes it Capping1.txt_random_random and so on. This disruption may lead to chaos in organizations.

The script made

```
REM TITLE: Automated File Renamer

REM AUTHOR: Bryan
REM DESCRIPTION: Renames files in a specified folder by appending "_random"
endlessly.

DELAY 1000
GUI r
DELAY 500
STRING cmd
ENTER
DELAY 1000
STRING cd %USER%\Documents
ENTER
DELAY 500
REPEAT
  STRING for /f "tokens=*" %%a in ('dir /b') do ren "%%a" "%%a_random"
  ENTER
  DELAY 1000
```

Strengths:

- It was an easy execution which leverages basic command prompt functionality.
- It is a non-destructive script and files can be restored to their original forms.
- Its greatest strength was disruption as it makes folders and files unmanageable.

Shortcomings:

- It needs folder access. The script can only work if the target system has little to no folder permissions. If the user has restrictions, the effectiveness of this script might be jeopardized.
- It is system specific. The script targets a specific folder path, in this case documents, and it must be correctly configured for this to work.

CAPS LOCK PRANK

Dependencies:

- PayloadStudio (DuckyScript)
- Access to the target system
- Windows OS with keyboard functionality enabled

Method of Delivery:

- Plug in the Rubber Ducky
- Activate the payload through pressing the internal button

Description:

This script was designed to frustrate the user by toggling the Caps Lock key on and off at random intervals. This disrupts the user's workflow when they are typing something. They unknowingly switch from lowercase to uppercase when typing either an email, an assignment or even a final report. Depending on the situation the user is in this script will prove to be very annoying. The randomness in the delays is what makes this script somewhat less predictable.

The script made:

```
REM TITLE: Caps Lock Prank (Infinite Loop)

REM AUTHOR: Bryan

REM DESCRIPTION: Continuously toggles Caps Lock to disrupt typing.


DELAY 1000

REPEAT

   CAPSLOCK

   DELAY 2000

   CAPSLOCK

   DELAY 1000

END_REPEAT
```

Strengths:

- It is both non-destructive. It does not cause any lasting or persistent changes/updates once stopped or removed.
- It is subtle annoying as the prank is not immediately realized but as time goes on the user then realizes what is going on.

Shortcomings:

- There is limited impact as this only occurs and can occur when the user is actively typing.
- It requires physical access to the system and this could make deployment very difficult in secured areas.

PHANTOM COMMAND PROMPT FLOOD

Dependencies:

- Payload Studio (DuckyScript)
- Windows Command Prompt
- Access to the target system

Method of delivery:

- Plug in the Rubber Ducky
- Activate the payload via the internal button on the RubberDucky
- Wait for the operation to reach its desired effect then remove the Rubber Ducky.

Description:

This is a more chaotic script as it simulates some form of system diagnostic process by opening multiple instances of the command prompt window. Each window displays messages like “Scanning System” or “Initiating process”. These windows appear very fast and repetitive. This payload remains active until the user terminates the process or reboots the entire system.

The script made:

```
REM TITLE: Phantom Command Prompt Flood
REM AUTHOR: Bryan
REM DESCRIPTION: Opens multiple Command Prompt windows with cryptic
messages.

DELAY 1000
REPEAT
   GUI r
   DELAY 500
   STRING cmd
   ENTER
   DELAY 500
   STRING echo Scanning system... && timeout /t 3 && echo Initiating process... &&
timeout /t 3
   ENTER
   DELAY 500
END_REPEAT
```

Strengths:

- The cryptic messages it leaves behind makes the less experienced user believe something very bad is going to happen or just gets them to panic.
- It is high in visual disruption as multiple windows open, it makes it difficult for users to navigate or focus.

Shortcomings:

- It has limited customizations. This script's effectiveness heavily relies on predefined messages which might not have the same impact on all users.

Key Note on scripts:

These scripts were made to highlight that simple yet effective automation can create a fun and harmless experience when handling a tool like the Rubber Ducky. These script contributions prioritize both fun and creativity, which proves that even with the simplest tools, when researched and used thoughtfully, can wreak havoc in a cybersecurity space.

<u>Bugs and Problems Encountered:</u>

Case Specific Bugs and Problems:

1. Microsoft Office Pop-up Interruption:
   - Using the Commands "INJECT_MOD Windows R" and "INJECT_MOD CTRL SHIFT ENTER" together to attempt to enter admin privilege mode

from the windows run window will sometimes cause Microsoft Office to pop up and break the script

  - To avoid this conflict, use “GUI r” + “CTRL SHIFT ENTER” as substitutes

2. Windows Defender blocking unauthorized .exe files
    - If trying to run a script injected by the RubberDucky causes the WindowsDefender to block it, try the following work around
    - Using windows Powershell in administrator mode, have the script input the following commands
        1. < STRINGLN Add-MpPreference -ExclusionPath “path to directory” >
        2. < STRINGLN Set-MpPreference -DisableRealTimeMonitoring $true >
    - These commands will set the path of the chosen directory to be ignored by the WindowsDefender, and for good measure, the followup command will disable the computer's ability to monitor for suspicious activity

Commons Bugs and Problems:

1. Opening windows in a script breaks it
    - If a script that involves opening multiple windows is in play and it seemingly breaks for no reason, add DELAY before key points in the script such as opening a window

    - NOTE: the required DELAY will differ based on the speed of the target computer, consider either researching targets RAM beforehand or giving a fairly generous DELAY length

---

DuckyScript Syntax:

Basic Syntax Keys:

REM:

- This is short for “Remark”, it is how you declare a comment
- REM This line is an example of a comment

STRING:

- Writes a string of characters
- STRING This line will be typed onto the target

STRINGLN:

- Works like a STRING however, presses the enter key at its end

VAR:

- Keyword is used to declare a variable
- Variable names begin with $
- VAR $<name> = <value>
- VAR $Num = 30

DEFINE:

- This keyword is used to declare a constant (like a variable but cant change)
- Constants do not begin with $
- DEFINE <name> = <value>
- DEFINE Num = 30

DELAY:

- Used to delay the next line in a payload by a set amount of milliseconds
- DELAY 500

Cursor Keys:

- UPARROW
- DOWNARROW

- LEFTARROW
- RIGHTARROW
- PAGEUP
- PAGEDOWN
- HOME
- END
- INSERT
- DELETE
- DEL
- BACKSPACE
- TAB
- SPACE

System Keys:

- ENTER
- ESCAPE
- PAUSE
- BREAK
- PRINTSCREEN
- MENU
- APP
- F1, F2, … F12

Modifier Keys:

- SHIFT
- ALT
- CONTROL
- CTRL
- COMMAND
- WINDOWS
- GUI

Lock Keys:

- CAPSLOCK
- NUMLOCK
- SCROLLLOCK

Device Setting Commands:

Buttons:

WAIT_FOR_BUTTON_PRESS:

- Halts payload until a button press is detected
- Not that if BUTTON_DEF has been declared at any point during the script, it will take precedence over this command

BUTTON_DEF:

- Used to define a function that is called when the button is pressed anytime within the payload as long as the button control is NOT already in use by the WAIT_FOR_BUTTON_PRESS command

  Example:

  ```
  REM When button is pressed, the string will be typed
  BUTTON_DEF
      STRING The button has been pressed
  END_BUTTON
  ```

DISABLE_BUTTON:

- Disables BUTTON_DEF from being called

ENABLE_BUTTON:

- Enables BUTTON_DEF to be called

LED:

- LED_OFF: Disables LED light
- LED_R: Enables red LED
- LED_G: Enables green LED

ATTACKMODE:

This setting will determine what device type the RubberDucky is emulating or pretending to be

- HID: Functions as a Human Interface Device (Keyboard)
- STORAGE: Functions as a storage device like a flash drive, use when copying files to/from the target
- HID STORAGE: Functions as both a HID and Storage device
- OFF: Will not function as any device, use when disconnecting from target

Optional Parameters:

- VID_: VendorID (16-bit HEX)
- PID_: ProductID (16-bit HEX)
- MAN_: Manufacturer (16 alphanumeric characters)
- PROD_: Product (16 alphanumeric characters)
- SERIAL_: Serial (12 digits)

Examples:

VID_023X = Vendor ID: 023X

PROD_RubberDucky = Product: RubberDuckey

SAVE_ATTACKMODE:

Saves currently running attackmode config

RESTORE_ATTACKMODE:

Loads previously saved attackmode config

Conditional Statements:

In DuckyScript like many languages, IF statements are the primary conditional statement. These are used to run blocks of code ONLY if a specific condition is met first

The syntax is as following:

```
IF (ConditionToBeChecked) THEN
        Code to run when
        Condition is met
END_IF
```

Operators:

Math Operators:

- = Assignment
- + Add
- - Subtract
- * Multiply
- / Divide
- % Modulus
- ^ Exponent

Comparison Operators:

Used to compare logical expressions in functions, IF statements, or loops

- == Equal to
- != Not equal to
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to

Logical Operators:

Used to connect logical expressions

- && - AND
- || - OR

Functions:

Functions are used for making blocks of reusable code.
Any DuckyScript can be used for the contents of a function

```
FUNCTION NameOfFunction()
        Contents
        Of
        function
END_FUNCTION
```

A function can then be called later in the code by typing its name, for example:

```
<<<<<
FUNCTION WriteAString()
        STRING This is a string
END_FUNCTION

WriteAString() — this line is what calls the function code
>>>>>
```

This code above will type out “This is a string”

Loops:

Determines a block of code that will run continuously until a condition is not longer True

```
WHILE (ConditionThatIsTrue)
    Code
    To be
    Looped
END_WHILE
```

An infinite loop can be created with the following syntax

```
WHILE TRUE
    Code
    To be
    Looped
END_WHILE
```

Bitwise Operators:

Operate on uint16 values at binary level

- & - Bitwise AND
    - If the bits of two operands are 1, results in 1
    - Else if either bit of an operand is 0, the result is 0
- | - Bitwise OR
    - If at least one bit of an operand is 1, results in 1

    - If none of the bits are 1, results in 0
- >> - Right Shift
    - Accepts two numbers, right shifts the bits of the first operand, the second operand determines the number of places to shift
- << - Left Shift
    - Accepts two numbers, left shifts the bits of the first operand, the second operand determines the number of places to shift

David’s Network Project:

Please click the link below to view this adjacent project’s documentation:

Network Project Documentation

This Concludes Our Documentation For The

<u>Hak5 - USB RubberDucky:</u>
The Keystroke Injection Tool

Thank you for Reading! :)