**Eshwar Reddy**

**Creating USB Rubber Ducky with Pendrive**

# Abstract

This project entails the construction of the USB Rubber Ducky which is a adversary tool that mimics a keyboard in order to automatically execute scripts on a target machine using a normal USB drive with autorun.ini configuration. Pre-installed scripts are triggered immediately on the insertion of the tool and they can include operating of applications, input of keystrokes, and even commands. These are USB Autorun Creator software, and Ducky Script Potential, and use this for penetration testing to check system weaknesses, to complete repetitive tasks and as an educational tool for encryption experts where they learn how to handle USB threats.

# Introduction

USB Rubber Ducky is a device that looks exactly like an ordinary USB drive, yet the moment it is plugged into the computer, it will act like a Human Interface Device displaying keyboard input, automatically carrying out pre-programmed commands. Although it was designed originally for penetration testing, it allows security professionals to look for vulnerabilities by automatically typing keystrokes that can open applications or run scripts in the absence of user permission. Projects for creating a USB Rubber Ducky using a normal USB drive, USB Autorun Creator, and a custom Ducky Script that runs on its own by simply inserting it. Devices of this nature are extremely useful for revealing system vulnerabilities, automating complex tasks, and increasing efficiency in systems-for example, IT administration, where they save time and also minimize human error through the automation of redundant tasks.

**Project Description**

# Technologies Used

To develop the USB Rubber Ducky, several tools, software, and hardware components were utilized to achieve the desired functionality. Below is a list of the key technologies involved:

1. **USB Autorun Creator**: This is actually a tool which helps to configure the USB drive in such a way that it starts running a certain script once connected. It generates an appropriate autorun.inf file which runs immediately when connected with a system.
2. **Ducky Script Language**: Ducky Script is a lightweight scripting language for the USB Rubber Ducky. It allows users to automate keystrokes among other actions, just like keyboard input. This can be used in command injection, opening applications, and interaction with the operating system.
3. **USB Drive**: A normal flash drive was transformed into a functional USB Rubber Ducky by uploading the autorun configuration and the Ducky Script onto the drive.
4. **Text Editor**: A simple text editor, like Notepad or Visual Studio Code, to create and make changes to the Ducky Script.
5. **Operating System (Windows)**: The main system on which the research, testing, and deployment of the USB Rubber Ducky was done, was a Windows machine. On this system, the behavior of the USB drive was tested to see if the autorun functionality worked as expected.

# System Workflow

The workflow of the USB Rubber Ducky-from creation right through to execution-is pretty simple yet effective. This is a summary of the whole workflow summarized below in steps:

1. **Script Development**: Writing of a Ducky Script aimed at automating some particular commands in a target machine; thus, it could range from opening a terminal to running system commands down to launching a web browser.
2. **USB Autorun Configuration**: After the script was written, the USB Autorun Creator software was used to configure the USB drive such that on insertion into a computer, it would automatically execute the Ducky Script. This tool produces an autorun.inf file which points to the Ducky Script and automates the running process without user interference.
3. **File Deployment**: The Ducky Script and autorun.inf file were downloaded onto the USB drive, making it a USB Rubber Ducky.
4. **Execution**: Insert this USB into a target computer and autorun starts running the Ducky Script. The computer detects this USB drive as a human input device or HID and acts out this script, just like a user has hit each key on the keyboard.
5. **Command Execution**: When the script executes, it will carry out the set of instructions predetermined in the script, which includes opening applications, manipulating system settings, or even injecting code. The system responds differently, depending on what real commands are encoded in the script.

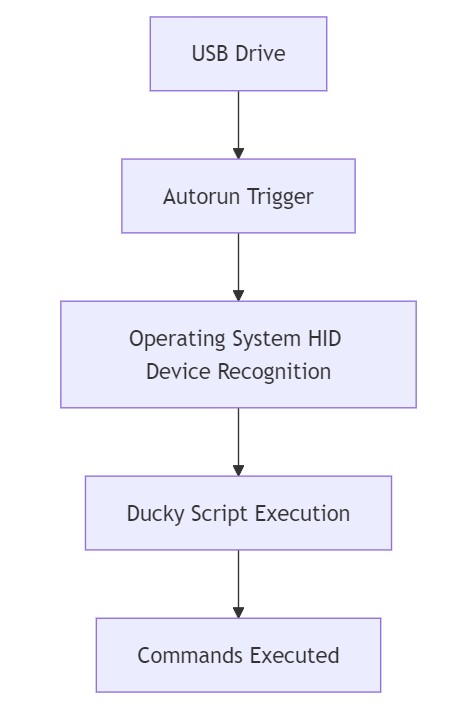
# System Architecture

The architecture of the USB Rubber Ducky system is relatively simple but powerful. The breakdown of the components and data and action flows is as follows in the system:

1. **USB Drive**: This hardware stores the autorun.inf file and the Ducky Script. It provides an entry point into the system.
2. **Operating System**: An OS of a targeted machine such as Windows will identify the USB device as an input peripheral and automatically run commands contained within the Ducky Script.
3. **Autorun Functionality**: An autorun.inf file residing on the USB initiates the running of the Ducky Script, bypassing any need for user interaction to run the script-a crucial ingredient to the smooth automation of commands.
4. **Ducky Script Execution**: This Ducky Script will start running upon a trigger from the autorun.inf file, emulating keyboard input to execute programmed commands onto the target machine.
5. **Command Execution on Target System**: The OS will run the script commands as if they were being typed in, which can also include opening applications, executing scripts, or changing system settings.

# System Architecture Diagram

Below is a conceptual diagram that represents the flow of data and actions between the USB drive, the operating system, and the execution of the Ducky Script.



* **USB Drive**: This hosts the Ducky Script and autorun configuration.
* **Autorun Trigger**: The script will automatically autorun through the autorun.inf once the USB is inserted.
* **Operating System**: Identifies the USB as a HID and thus allows keystroke simulation.
* **Ducky Script**: Actually, performs the set of predefined commands on the system.
* **Commands Executed**: This is the last step where the operating system performs the actual execution of the actions scripted in the Ducky Script.

# Manual for Use and Setup

The following section will detail how to set up and use the USB Rubber Ducky by installing the necessary software, writing and deploying the custom Ducky Scripts on a standard USB drive, and testing the setup and password protection considerations.

# Step 1: Preparing the USB Drive

1. **Select a USB Drive**: An average USB flash drive would suffice but of a size that will accommodate this, at least 1 GB free space.
2. **Format the USB Drive**:
   * Open your **File Explorer**. o Right-click your USB drive in the menu that pops up and click Format. o Make sure the File System is Fat 32 for better compatibility.
   * Click Start to format the USB drive. All data will be deleted, so make sure you have everything backed up.

# Step 2: Installing the USB Autorun Creator

1. **Download the Software**:
   * Access the official website of USB Autorun Creator or any trusted site that offers this tool.
   * Download the installation file and follow prompts for installation onto your system.
2. **Install the Software**:
   * Run the installation file.
   * Follow the on-screen instructions that pop up for installation.
3. **Launch USB Autorun Creator**:
   * After installation, open USB Autorun Creator. This utility you are going to use for setting up autorun on your USB.

# Step 3: Writing the Ducky Script

1. **Understanding Ducky Script**:

o Ducky Script: A simple basic script used with a rubber ducky device to emulate keystrokes. Every line of the script corresponds to something the keyboard will do from press to open applications.

Example Ducky Script:

vbnet

Copy code

DELAY 500

GUI r

DELAY 100

STRING notepad

ENTER

DELAY 500

STRING This is a test script.

ENTER

2. **Create the Script**:

o Open a text editor (such as **Notepad** or **Visual Studio Code**).

Write the commands you want the USB Rubber Ducky to execute when plugged in.

This is the Script I created.

@echo off

REM Create a text file named hello.txt with the message "H A C K E D"

(

echo ## ## ###### ####### ## ## ###### ###### echo ## ## ## ## ## ## ## ## ## ## echo ## ## ## ## ## #### ###### ## ## echo ## ### ## ######## ## ## ## ## ## ## echo ## ## ## ## ## ## ## ## ## ## echo ## ## ## ## ########## ######### ######

) > hello.txt

REM Open the text file using Notepad (or your default text editor) start notepad hello.txt

REM Exit the script

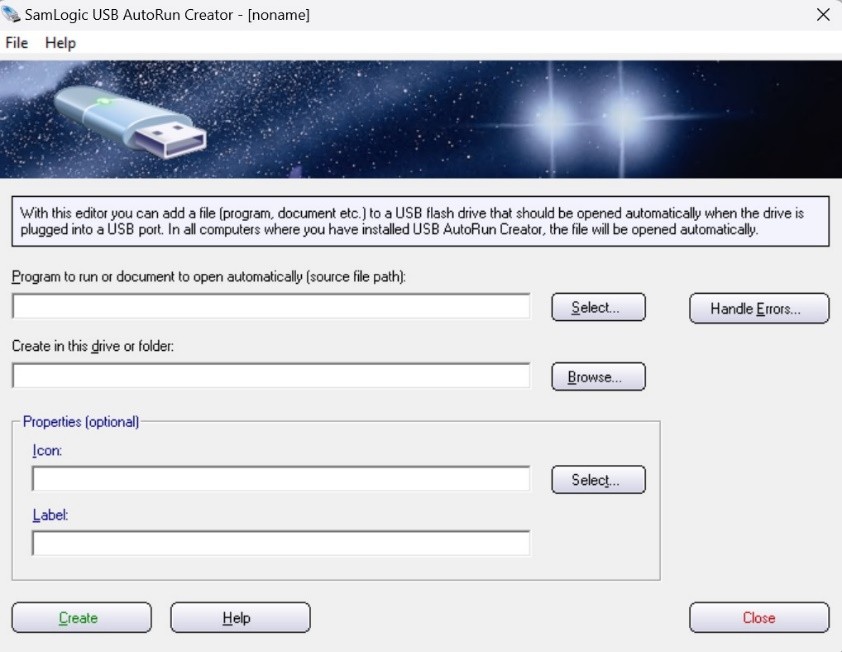
Exit

3. **Save the Script**:

* Save your script as a plain text document and give it the extension .bat. Example: ducky\_script.bat.
* Remember where you save the file, because you'll need to link to it in the following step.

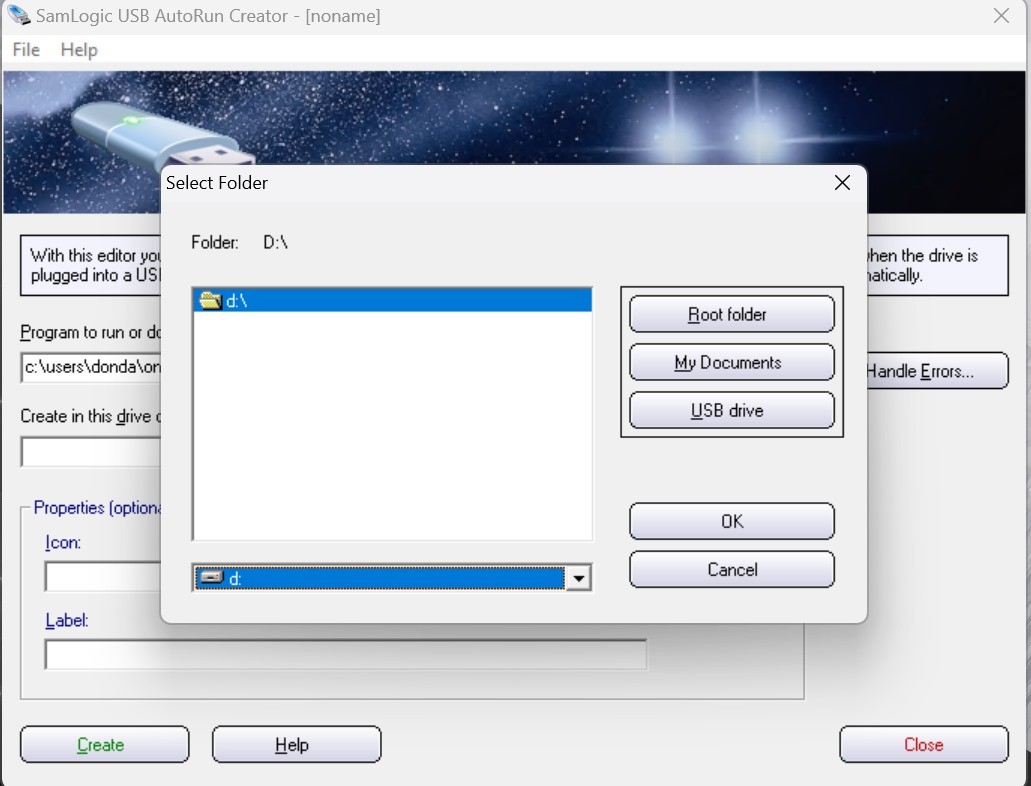
# Step 4: Configuring the USB Autorun

1. **Open USB Autorun Creator**: o Open the USB Autorun Creator software with the USB plugged in.

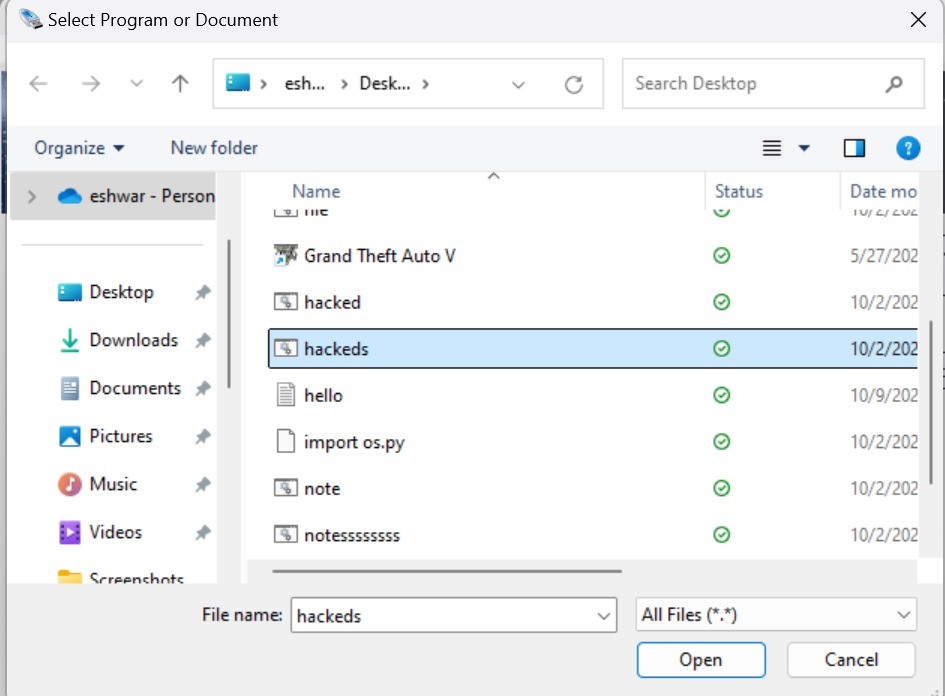


1. **Select the USB Drive**:

o In the main interface, select your USB drive among the listed ones.

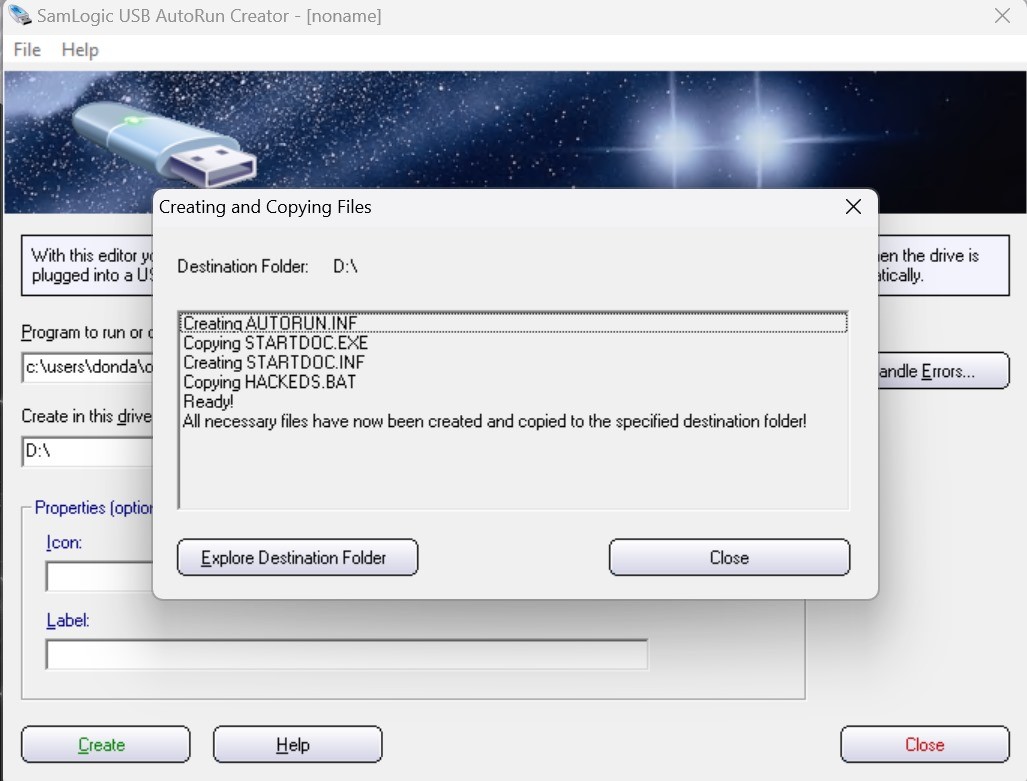


1. **Add the Ducky Script**:
   * Navigate to the option where you can configure the autorun.
   * Set the path to the Ducky Script: ducky\_script.bat, created earlier.
2. The tool will now create the required autorun.inf file for you linking to your script.



1. **Create Autorun**:

o Click on Create or Build to create the autorun files onto your USB drive. It copies the autorun.inf and your Ducky Script onto the drive.



# Step 5: Testing the Setup

1. **Insert the USB Drive**:
   * Insert the USB drive into the target computer. Make sure autorun is enabled on the machine for the script to run automatically.
2. **Observe the Script Execution**:
   * When the autorun file kicks in, the Ducky Script should automatically run. If you programmed the script to, say, open Notepad and type a message, you should immediately see those specific actions take place as soon as the USB is inserted.
3. **Troubleshooting**:
   * By the long shot, if this does not work, ensure autorun is on the target computer. o Ensure that the autorun.inf is set correctly to point towards your Ducky Script. o If necessary, manually execute the script by opening the USB drive and running the file.

# Step 6: Customizing the Ducky Script

1. **Modifying Commands**:
   * You may change what the Ducky Script does at any time, so you can have it do whatever you want. For example, you may edit this script to open a browser and/or open command prompt or start an application.
2. **Testing New Scripts**:
   * After you edit your script, proceed again to Step 4, making sure that you copy the new modified script to your USB drive and test the script as needed.

# Passwords and Security (Optional)

* **Password Protection**: You can, if you want, password-protect the files on the USB using thirdparty software encryption applications such as BitLocker or VeraCrypt. This is an optional step; basically, for scenarios when you don't want unauthorized people to access the script files.
* **Credential Prompt**: Depending on what your script performs, you may want to prompt the script for credentials, or request the user's input. For example, if the script does something administrative, you can make a prompt to ask the user for a password so it allows the execution.

# Notes for Use

* **Compatibility**: The target machine should have autorun enabled. Autorun has been disabled by default in a number of operating systems for security reasons, mostly on newer operating systems like Windows.
* **Ethical Use**: The USB Rubber Ducky should only be used for lawful purposes, including penetration testing or automating tasks that may be regarded as legitimate. Using such a device to breach somebody's security or privacy is against the law.

# Problems Faced and Unresolved Issues

The project with the USB Rubber Ducky came with several challenges that affected its functionality and usability, which included:

* **Compatibility Issues with Modern Operating Systems**: Windows 10 and Windows 11 normally disable autorun via security updates. This is likely to make the automatic running of scripts very difficult. Doing this manually did indeed decrease the effectiveness of automation. One considered solution was enabling autorun with Group Policy, but this was not the most viable long-term solution.
* **Cross-Platform Incompatibility**: Heavy limitation for running Autorun on macOS and Linux, whereby the script has to be run manually, hence reducing cross-operating system compatibility for this project.
* **Script Debugging and Error Handling**: It is difficult to debug the script since Ducky Script does not have debugging tools. Since there were no error messages provided when a script failed running because of syntax mistakes, it was hard to find errors.
* **USB Recognition as a Human Interface Device (HID)**: Incomplete recognition as a keyboard HID by some systems lowers its reliability since the script will not execute if recognition does not take place.

# Future Improvements

Following are some improvements that can be made for better functionality, security, and usability of the USB Rubber Ducky. Some of the most important key future improvements include:

1. **Encryption for Security**: Adding encryption either to the Ducky Script or to the whole USB would protect sensitive scripts from unauthorized access. Full-disk encryption can be done using BitLocker or VeraCrypt.
2. **Cross-Platform Support**: Increased versatility could come from developing support for macOS and Linux; this can be provided through platform-specific executables or OS detection.
3. **User-Friendly Interface**: Furthermore, a GUI for customizing scripts with prebuilt templates will make things easier on the user side since not everyone has technical skills.
4. **Expanded Script Functionality**: Conditionals and input statements would make the scripts even more interactive and flexible to use in a variety of situations.
5. **Improved Device Detection**: The pursuit of greater reliability of the USB as a Human Interface Device across systems for consistency in device detection and execution of scripts..

These improvements would make the system more robust, secure, and accessible for a wider range of users and environments.

# Conclusion

A USB Rubber Ducky was constructed for this project, which can be programmed to automatically run through any series of scripts upon being inserted into a computer, utilizing autorun functionality and the Ducky Script emulating keyboard input. The system successfully opens applications and runs scripts, finding its perfect fit in tasks related to penetration testing and administrative automation. The project has shown understanding in the security risks with USB and finds practical use in education in cybersecurity and IT administration.

The basic technologies to be utilized were USB Autorun Creator, Ducky Script, and the Windows OS as a test environment. For instance, the USB Rubber Ducky had some strong limitations, working on most modern operating systems at the autorun stage, which would limit how much automation you could carry out because some functions could not support cross-operating system basis. All the same, the project showed a lot about how USB-based automation can be affected and where this can go. Future enhancements shall seek to make the system more secure, more cross-operable, and friendly for use by users.