

AL-ZAYTOONAH UNIVERSITY OF
JORDAN

Cybersecurity Department

Graduation Project

Amman, Jordan



Project title

Connecting keylogger with esp32

And detect from these attacks

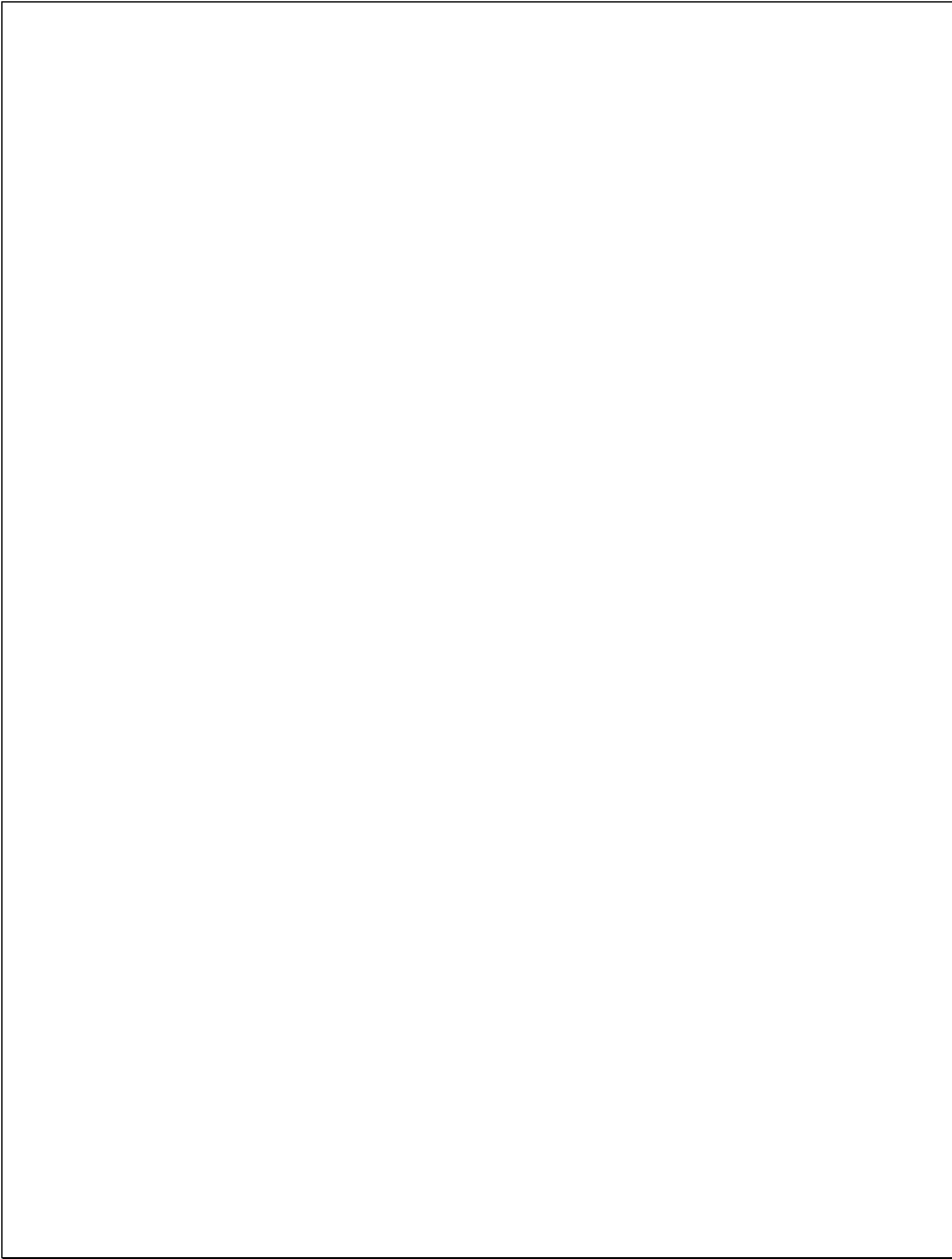
Prepared by

Khaled Jamal Haimur

Supervised by

Dr. Bilal Hawasheen

June 2024



Abstract

This project aims to explore the danger of keystroke logging using simple devices such as the ESP32, and highlight the threats these devices pose to cybersecurity and privacy of individuals and companies. The project examines how to map keystrokes to an ESP32 device and monitor the resulting activity, revealing how easily the privacy of individuals and organizations can be compromised by cheap, readily available hardware.

The project was implemented by designing an ESP32 based system to monitor and record keystrokes from connected devices. This was achieved by programming the ESP32 to act as an interface between the keyboard and the computer, picking up keystroke signals and sending them wirelessly to another device to monitor. The results showed that these devices are capable of intercepting and recording sensitive data with high efficiency, highlighting the urgent need to adopt strong security measures.

This type of attack targets a wide range of people, including individuals who use computers for personal or professional purposes, companies that handle sensitive data, and government organizations that need to protect their information. The seriousness of these attacks lies in the violation of privacy and theft of financial and personal data, exposing individuals and companies to financial and reputational losses.

Abstract

The project also focuses on possible solutions to counter these threats, emphasizing the importance of data encryption to protect sensitive information from being hacked. The project discusses encryption mechanisms and best practices for securing communications, including the use of advanced encryption techniques such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman). In addition, the project provides recommendations to enhance awareness of the importance of cybersecurity to maintain privacy, such as training employees to recognize cyber threats and adopting strict security policies within organizations.

Other recommendations include using antivirus software and updating it regularly, ensuring secure networks are used when handling sensitive data, and adopting identity and access management (IAM) solutions to ensure that only authorized people have access to sensitive information.

key words

Cybersecurity, keystrokes, ESP32, data encryption, privacy, information protection, identity and access management, cybersecurity training.

Acknowledgement

I would like to express my deepest gratitude to all those who have contributed to the completion of this graduation project. Firstly, I extend my sincere appreciation to my supervisor [dr. Bilal Hawasheen], and To our Cyber security department teachers and Dr, additionally Dr. Ahmad Alkhateeb for their invaluable guidance, continuous support, and insightful feedback throughout the duration of this project. Their expertise and encouragement have been instrumental in shaping the direction of this research.

I would also like to thank my family for their unwavering love, encouragement, and understanding throughout my academic journey. Their support has been my source of strength and motivation.

Additionally, I extend my gratitude to my friends and colleagues for their encouragement, assistance, and valuable discussions, which have enriched the development of this project.

Finally, I am grateful to the faculty members and staff at [Al- Zaytoonah university of Jordan/cyber security department] for providing the necessary resources and conducive environment for academic and research pursuits.

Table of content

1. Introduction

Background.....	1
Problem Statement.....	1
Significance of the Study.....	1
Objectives.....	1
Methodology.....	2
Expected Results.....	2

2. literature review

2.1 ESP32.....	3-4
2.2 ESP32 pins.....	5-6
2.3 Potential security challenges in using ESP32.....	7
2.4 OWASP IOT.....	8-9
2.5 Keystrokes.....	10-11
2.6 Encryption.....	12-13

3. Methodology

3.1 Introduction.....	14
3.2 System description.....	15
3.2.1 Explain.....	15
3.2.2 ESP32 programming using C++	
3.2.2.1 Arduino Setup.....	15,16,17
3.2.2.2 An explanation of the functions and libraries used in programming.....	17,18,19
3.2.3 programing keylogger using python.....	20,21
3.3 Procedures	
3.3.1 Using buttons in ESP32.....	23
3.3.2 Environment Setup.....	24
3.3.3 Possible problems in the project.....	25,26,27
3.3.4 Detecting from keylogger.....	28

4. Experiments

4.1 Performance testing and evaluation.....	29,30,31,32
---	-------------

5. Conclusion.....33,34

1.Introduction

1.1 Background

Keystroke logging is a technology that has long been used in a variety of fields, ranging from legitimate uses such as performance analysis and security improvement, to illicit uses such as espionage and data theft. In recent years, simple devices like the ESP32 have become more popular due to their ease of use and low cost, making them effective tools in the hands of cyber attackers.

1.2. Research problem (Problem Statement)

Keystroke monitoring using simple devices like the ESP32 is a serious threat to the cybersecurity and privacy of individuals and businesses. The danger of these devices lies in their ability to intercept and record sensitive data with high efficiency, which exposes individuals and companies to financial losses and bad reputation as a result of privacy violations and theft of financial and personal data.

1.3 Significance of the Study

This study gains great importance in light of the increasing threats to cybersecurity. Understanding how simple devices like the ESP32 can be used to monitor keystrokes helps develop effective strategies for protecting data and preserving the privacy of individuals and businesses. This study can contribute to enhancing awareness of the importance of cybersecurity and encouraging the adoption of better security practices.

1.4 Research objectives

Explore how keystroke monitoring works using the ESP32.
Analyze the risks and threats resulting from this technology.
Develop strategies to encrypt data and protect it from these attacks.
Provide recommendations to improve cybersecurity practices and reduce exposure to these threats.

Introduction

1.5 Research Methodology

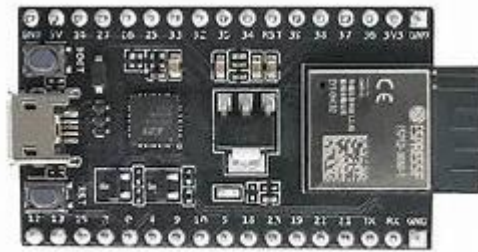
This project is based on designing a system using an ESP32 device to monitor and record keystrokes. This was achieved by programming the ESP32 to act as an interface between the keyboard and the computer, picking up keystroke signals and sending them wirelessly to another device to monitor. The resulting data will be analyzed to determine the effectiveness of these attacks. Advanced encryption technologies such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) will also be tested to evaluate their ability to protect data from interception and espionage.

1.6 Expected Results

The results are expected to show that devices such as the ESP32 can easily be used to monitor keystrokes and should therefore provide adequate security solutions to prevent this type of attack. The results are also expected to highlight the importance of educating users and implementing cybersecurity policies in companies and institutions.

2. literature review

2.1 Esp32



ESP32 is a general term that refers to a set of developer-friendly development boards and chips that have Wi-Fi capabilities. These devices are based on technology from Espressif, a Chinese chip manufacturer founded in 2008. Its first product, a 2.4 GHz Wi-Fi system chip (SoC), came to market in 2013 under the name ESP8089, and was intended for hardware applications Tablets and TV boxes. However, it was the ESP8266, introduced in 2014, that caught the attention of the developer community.

Espressif's vision is focused on bringing modern Artificial Intelligence of Things (AIoT) solutions to market through wireless technology capable of saving energy. The ESP8266 brings together these threads by providing an easily usable single-chip device with the software needed to communicate over Wi-Fi networks.

Basically, the ESP8266 is powered by a Tensilica Xtensa L106 32-bit RISC processor. This rare vertical architecture is offered as licensable intellectual property by Cadence and, according to a press release from 2007, provides higher MIPS performance in Dhrystone than the Arm Cortex-M3 processor. Furthermore, they claim that the core operates at a rate of less than one mW per MHz, which is attractive when dealing with battery-powered IoT applications.

2.1 ESP32

Basically, the ESP8266 is powered by a Tensilica Xtensa L106 32-bit RISC processor. This rare vertical architecture is offered as licensable intellectual property by Cadence and, according to a press release from 2007, provides higher MIPS performance in Dhrystone than the Arm Cortex-M3 processor. Furthermore, they claim that the core operates at a rate of less than one mW per MHz, which is attractive when dealing with battery-powered IoT applications.

The developer community became the first users of these devices when Hackaday announced in 2014 the launch of a new \$5 Wi-Fi module sold through Seeed Studio. Known as the ESP-01, this module was made by third-party manufacturer Ai-Thinker, also based in China. With only eight pins, including one that provides a UART interface for controlling the module via AT commands, the module was easy to integrate with the Arduino platform, providing Internet connectivity for simple modules like the Arduino Uno.

The only point that was a small hitch: all documents were written in Chinese. However, this was not a major obstacle for the developer community as volunteers began translating the available documentation using Google Translate and writing software libraries.

Regarding the cheap cost of the ESP32, the beauty of the ESP8266-based modules (Figure 1) lies in the limited number of components required to build a functional Wi-Fi module. Assuming a 3.3V supply is available, the designs need only a few resistors and capacitors, a PCB antenna, an external QSPI serial flash memory, and a crystal operating between 24 and 52 MHz. When powered on, the ESP8266 extracts the firmware from the flash and then copies it to the internal SRAM where it is executed.

-Benefits of connecting Keylogger to ESP32

Local Storage: The ESP32 memory can be used to store data collected from the keylogger, either in the form of a microSD card or in internal memory.

Intelligent Control: The ESP32 can be programmed to turn the keylogger on and off or to adjust its settings and control it remotely, allowing its functions to be customized and adapted to specific needs.

Wireless Communication: The ESP32 can be used to connect the keylogger to other devices via Bluetooth or other wireless communication protocols, allowing it to transfer data seamlessly without the need for wired connections.

Power Control: The ESP32 can effectively manage the power consumption of the keylogger, which can help improve battery life or reduce power consumption when using other power sources.

2. literature review

2.2 ESP32 pins

The ESP32 is a powerful and versatile microcontroller board widely used in IoT and embedded systems projects. Understanding the ESP32 pinout, which refers to the arrangement and functionality of the GPIO (General Purpose Input/Output) pins, is essential for effective project development.

The ESP32 has a variety of pins that provide different interfaces for communicating with external components such as electronic devices, sensors, displays, etc.

Digital Pins:** These pins are used to control digital signals, as they can be in a high or low state. Often used to connect digital devices such as LED lights and sensors.

Analog Pins:** These pins are used to measure analog signals such as changes in voltage or current. Typically used with analog sensors such as temperature sensor.

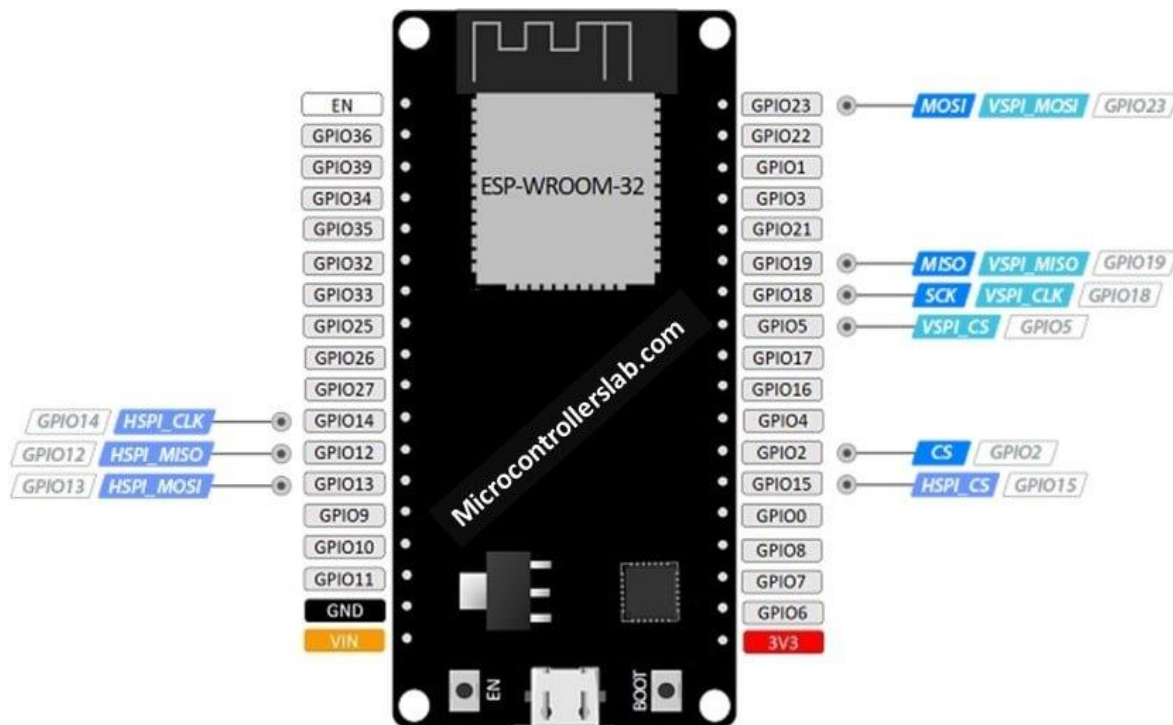
GPIO Pins (Programmable Input and Output Pins):** These pins are used to specify their functions as inputs or outputs by programming. They can be used to connect different devices and controlled by software code.

Power Pins:** These pins are used to connect power to the circuit, such as ground and positive power (Vcc).

5. Communication Pins: They are used to communicate with other devices via standard interfaces such as UART, SPI, and I2C.

2.2 ESP32 pins

The use of these pins allows control and communication between the ESP32 and external components, expanding the possibilities for development and interaction with many different applications in the field of Internet of Things (IoT) and embedded electronics.



2.3 Potential security challenges in using ESP32

Using the ESP32 in keystroke monitoring projects may face some potential security challenges, including:

Software and hardware vulnerabilities: The software used in the ESP32 may contain vulnerabilities that could allow an attacker to gain unauthorized access to logged data or control of the device.

Hardware attacks: The ESP32 itself may be vulnerable to hardware attacks that may affect its performance or allow unauthorized access to data.

Espionage and exfiltration: Attackers may use compromised ESP32 devices as espionage and exfiltration tools, putting sensitive data at risk.

Data loss: Data loss may occur due to problems storing data on the ESP32 or transferring it insecurely.

Wireless Threats: The ESP32 may be vulnerable to wireless attacks such as data interception during wireless transmission or unauthorized access to connected Wi-Fi networks.

Advanced Threats: The ESP32 may face advanced threats such as phishing attacks and malicious attacks that may directly target the system.

“Among the potential security challenges in using the ESP32 in our project comes communicating with OWASP for IoT concepts. The OWASP concept of IoT security refers to a set of guidelines and practices that aim to improve the security of devices and applications connected to the Internet. By analyzing these concepts and applying them to our project, we realize The importance of identifying and addressing potential security threats.

2. literature review

2.4 OWASP IOT

For example, OWASP for IoT analysis shows that communication between Internet-connected devices can be a source of security threats such as unauthorized intrusion and data tampering. Therefore, we must take necessary measures to secure ESP32 communications and protect the data that is exchanged between it and other devices.

“Using the OWASP for IoT guidelines as a framework for analyzing security threats, we can implement effective security measures such as data encryption and implementing validation mechanisms to ensure the integrity of communications and protect against external threats.”

OWASP is short for "Open Web Application Security Project", an international non-profit community that works to improve web application security. Founded in 2001, OWASP includes a group of professionals, developers, and curators working to provide knowledge and tools to improve web application security.

OWASP for IoT is part of the OWASP project's efforts to enhance the security of Internet-connected systems, which include smart devices, Internet-connected devices, and more. OWASP for IoT aims to provide guidance and resources for examining, evaluating, and improving the security of IoT devices.

OWASP for IoT addresses a variety of security challenges facing IoT devices, including:

External Threats: The main challenge in IoT security is external threats such as hacking, espionage, and data tampering, which can lead to exploiting vulnerabilities in devices and exposing data to risk.

2.3 OWASP IOT

Weak security design: There may be weaknesses in the design of smart devices and associated software, resulting in security vulnerabilities that allow unauthorized access or tampering with data.

Insider Threats: Security threats can arise from within the network, such as insider attacks and breaches resulting from inattention or fraud.

By applying OWASP principles for IoT, developers and engineers can improve the security of their devices and applications and reduce security threats. OWASP for IoT provides guidance, tools, and resources to help the community understand and address the security challenges facing their Internet-connected systems.

2.5 keystrokes

“Keystrokes” is a term that refers to keystrokes or buttons on a computer keyboard. When a user presses a key on the keyboard, an electronic signal is generated that represents the letter or symbol associated with that key. These records records signals or "presses" that contain information about when and which button was pressed, and can be used to track activities on the computer.

The use of keystrokes in cybersecurity and academic research typically focuses on studying the pattern of keyboard strokes to identify suspicious activity or intrusions. For example, keystroke analysis can be used to identify unauthorized login attempts, detect phishing attacks, or monitor users' behavioral activity.

Old research on keystrokes usually focused on developing techniques for analyzing and detecting unauthorized activity using keystroke recording. There are many studies conducted by universities and research centers that have explored how to use “keystrokes” in various fields such as information security and learning about human behavior.

Examples of these studies include:

A study conducted by the University of California, San Diego, explored how keystroke analysis can be used to detect phishing and targeted intrusions.

Research conducted by Stanford University used keystroke analysis to determine users' behavioral activity in business environments and detect internal intrusions.

2.4 keystrokes

A study conducted by the University of Massachusetts developed new techniques to analyze and classify keystrokes more efficiently and accurately.

These studies and research highlight the importance of analyzing “keystrokes” in the fields of cybersecurity and identifying human behavior, and contribute to developing technologies and tools to combat cyber threats.

2.6 Encryption

Encryption concept

Encryption is the process of converting information from a readable form to an unreadable form using complex mathematical algorithms, so that the encrypted information can only be read by authorized persons, who possess the key necessary to decrypt it. Encryption is a vital part of information security and is used to protect data from unauthorized access, whether this data is transmitted over networks or stored in storage devices.

The importance of encryption

Privacy Protection: Encryption ensures that personal data and sensitive information remain confidential and cannot be accessed by unauthorized persons.

Ensuring security: Encryption helps protect data from tampering or modification while it is being transmitted over networks. It ensures data integrity and prevents attackers from changing its contents.

Keylogger and encryption

Importance of encryption in keylogger

Using a Keylogger involves recording keystrokes, a process that may involve collecting sensitive information. It is therefore necessary to encrypt data to protect it from unauthorized access when it is transferred over networks or stored.

KeyScrambler

What is KeyScrambler?

KeyScrambler is a type of security software designed to protect keystrokes from malicious software (such as keyloggers). KeyScrambler works by encrypting keystrokes as they are entered into the keyboard, and only decrypting them when they reach the target application, making it difficult for malware to intercept and decrypt these keystrokes.

How KeyScrambler works

Encryption at source: When a user presses a key, KeyScrambler encrypts that keystroke before it reaches the application the user is using (such as a web browser or text program).

Secure transmission: Encrypted keystrokes are transmitted throughout the system, making it difficult for spyware to intercept and read them.

Decryption at destination: When keystrokes reach the target application, KeyScrambler decrypts them so that the application can process them correctly.

Benefits of KeyScrambler

Malware Protection: KeyScrambler protects users from keyloggers that attempt to intercept keystrokes to steal sensitive information such as passwords and personal information.

Ease of use: KeyScrambler runs in the background without requiring constant user interaction, making it a practical and simple solution for enhancing system security.

Comprehensive Security: KeyScrambler adds an extra layer of security to traditional security tools such as antivirus software and firewalls.

3 Methodology

3.1 Introduction

In this chapter, we will review the methodology we followed to achieve the main goal of our project: linking the ESP32 module to a keylogger to improve data security and better understand users' behavior. This project aims to develop a system that can monitor and record keystrokes and send this data to a central processing unit via an ESP32 module. To achieve this goal, a comprehensive and detailed methodology was adopted to ensure accurate data collection and analysis in a way that allows reliable results to be reached.

Below, the experimental design of the study, procedures used to collect data, measurement tools used, study sample, data analysis methods, as well as ethical considerations associated with the research will be detailed. Limitations that may affect the research results and how to address them will also be discussed. This methodology aims to provide a reliable framework for replicating the study in different settings and obtaining consistent results.

3.2 System description

3.2.1 Overview

Our system consists of two main parts: the Keylogger developed using Python, and the ESP32 module programmed using C++. The system records keystrokes and sends them to the ESP32 unit, which in turn processes the data and sends it to the server or final storage unit.

3. Methodology

3.2.2 ESP32 programming using C++

3.2.2.1 Arduino Setup

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's commonly used by hobbyists, students, and professionals to create interactive projects and prototypes.

Arduino IDE (Integrated Development Environment):

The Arduino IDE is a software application that provides an easy-to-use interface for writing, compiling, and uploading code to Arduino boards. It's available for Windows, macOS, and Linux.

Getting Started:

1.Install Arduino IDE: Download and install the Arduino IDE from the official website (<https://www.arduino.cc/en/software>).

3. Methodology

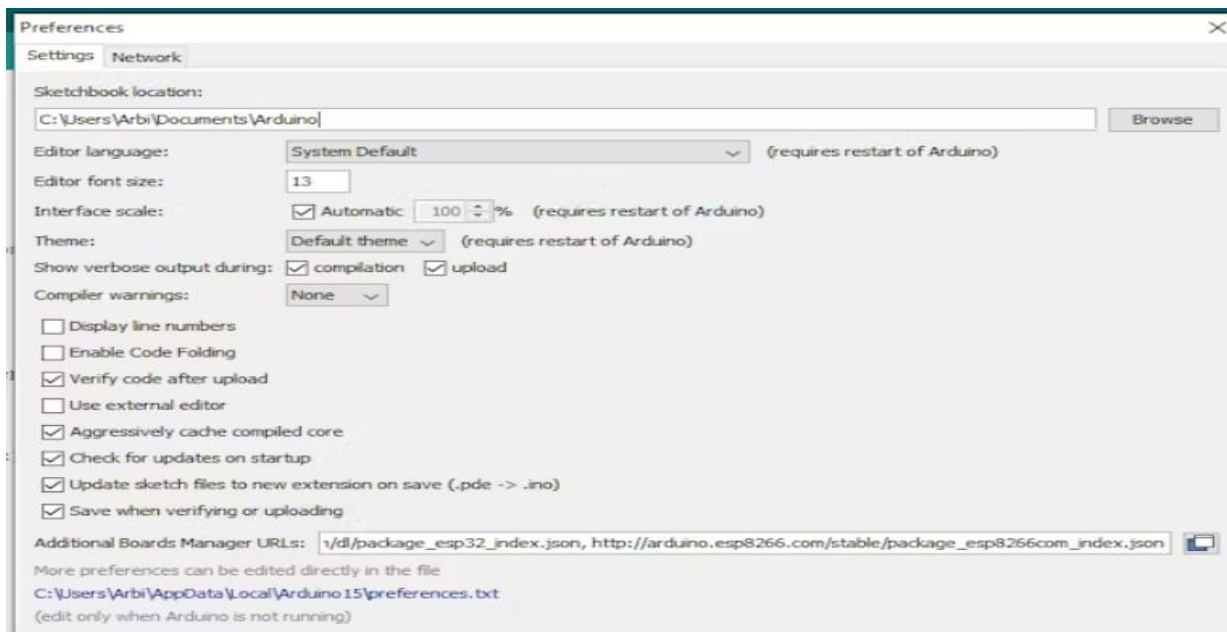
3.2.2.1 Arduino Setup

2.Connect Arduino Board: Connect your Arduino board to your computer using a USB cable.

(before select board we need esp32 preference link)

https://espressif.github.io/arduino-esp32/package_esp32_index.json,

https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json



Library links in an integrated development environment (IDE) provide an important advantage to developers for several reasons:

Easy to access: By having links in the IDE, developers can easily access the libraries they need without having to search long online.

Save time: Developers can save time searching for appropriate libraries online by using direct links in the IDE.

Firmware updates: May include links to the IDE's installed software store, which can provide updates to libraries on a regular basis, allowing developers to take advantage of newer versions and improvements without the hassle of manual updating.

3.Select Board: In the Arduino IDE, go to Tools > Board

and select the appropriate Arduino board you're using

Example (ESP32 by Espressif).

4.Select Port: Also in the Tools menu, select the port to which your Arduino board is connected.

3.Methodology

3.2.2.2 An explanation of the functions and libraries used in programming

The Sketch:

In Arduino programming, a program is called a "sketch." Here are some key components of an Arduino sketch:

1.Setup () Function: This function runs once when the Arduino board is powered on or reset. It's used for initializing variables, pin modes, etc.

2.Loop () Function: This function runs repeatedly as long as the Arduino board is powered on. It's where the main logic of your program resides.

3.Comments: You can add comments to your code using // for single-line comments or /* */ for multi-line comments. Comments are ignored by the compiler and are used to explain the code.

3. Methodology

3.2.2.2 An explanation of the functions and libraries used in programming

4. Variables: Variables are used to store data. They can be of different types such as int, float, char, etc.

5. Functions: You can define your own functions to modularize your code and make it easier to understand.

6. Libraries: Arduino provides a vast collection of pre-written code called libraries, which you can use to interface with various sensors, actuators, displays, etc.

Writing Code:

1. Write Your Code: Use the Arduino IDE's text editor to write your code. You can write code to read sensors, control actuators, process data, and more.

2. Verify/Compile: Click the checkmark icon to verify your code. This checks for syntax errors and compiles the code into machine-readable instructions.

3. Upload: Once your code compiles successfully, click the right arrow icon to upload it to the Arduino board.

4. Monitor: You can open the Serial Monitor (Tools > Serial Monitor) to view messages sent from your Arduino board for debugging purposes.

WiFi.h: This library provides the functionality needed to set up and manage a Wi-Fi connection using an ESP32 device. These functions include creating an Access Point or connecting to an existing Access Point.

WiFiClient.h: This library provides functions to create a TCP/IP client to communicate with TCP/IP servers over a Wi-Fi network. They can be used to send and receive data to and from a specific server.

WiFiServer.h: This library provides functions to create a TCP/IP server on an ESP32 device, allowing other devices to connect to it and exchange data.

Elementary functions and configurations:

First, the necessary libraries are called and the variables necessary to set up the network connection are defined.

The ssid and password variables are known to store the name and password of the Wi-Fi network.

Creates a WiFiServer object named server listening on port 12345.

Setup () function:

Called once when the program starts.

Serial communication is configured to communicate with the host computer.

Connects to the selected Wi-Fi network using the given network name and password.

The Server is started to receive incoming connections.

loop() function:

It is run continuously after the setup() function has finished and is used to execute commands permanently.

A new client is created when a new connection is received on the server.

It checks for the existence of a client and if there is, reads the data sent by the client.

Prints data received via serial communication.

This code basically creates a Wi-Fi access point on the ESP32 device, and when that access point is connected, a TCP/IP server is started listening on port 12345. Whenever a device connected to the server sends data, it is read and printed via serial communication.

3.2.3 programming keylogger using python

Keyloggers are a particularly insidious type of spyware that can record and steal consecutive keystrokes (and much more) that the user enters on a device. The term keylogger, or “keystroke logger: Software that logs what you type on your keyboard.

Import libraries:

The pynput.keyboard library is imported for keyboard monitoring.

The socket library is imported to establish a TCP/IP connection.

Network connection setup:

The IP address of the ESP32 device is known.

The port number is known for communication with the ESP32 device.

Establishing the TCP/IP connection:

The socket object is created using the socket module.

The connection to the ESP32 device is made using its previously defined IP address and port number.

Definition of functions:

on_press(key): Called when a key is pressed on the keyboard. The CD character or key name is sent to the ESP32 device.

on_release(key): Called when the key is released. If the key is "esc", the program is stopped.

Start listening for keystrokes:

Keyboard listening is initiated using the on_press and on_release functions.

Listening is run in a “with” loop to ensure the listener closes properly when finished.

Close connection:

The TCP/IP connection to the ESP32 device is closed when the program finishes. This code monitors the keyboard and sends keystrokes to the ESP32 device over a TCP/IP connection,

allowing the device to control certain operations based on the keystrokes sent.

3.3 System description

3.3.1 Using buttons in ESP32

1. Button 1

the description:

This button is commonly known as the "BOOT" or "GPIO0" button.

It is usually connected to the GPIO0 pin on the ESP32 module.

Basic functions:

System Reset: When the button is pressed simultaneously with the unit restarting, it can reset the system.

Enter Flash Mode: This button is commonly used when connected to a PC to program the unit. The BOOT button must be pressed during reboot or when the unit is connected to the computer to activate programming mode.

Custom Functions: This button can be programmed to perform specific tasks, such as starting data logging, sending data, or changing system settings. Its function can be customized programmatically using C++ codes to suit the needs of the project.

2. Button 2

the description:

This button is usually known as the "EN" or "RST" button.

It is usually connected to the EN pin on the ESP32 module.

Basic functions:

Reset: When the button is pressed, the unit is restarted (reset) regardless of the current operating status. This is useful for quickly restarting the unit during development or if an error occurs.

Custom Functions: This button can be programmed to perform other specific tasks, such as stopping data logging, changing the operating mode, or sending a specific signal to another part of the system. Its function can be customized programmatically using C++ codes.

3.3.2 Environment Setup

Detailed procedures for implementing a project to connect ESP32 to Keylogger

Device 1

Hardware used: ESP32 module, USB cable to connect ESP32 to computer, suitable power supply for ESP32, connect ESP32 module to computer using USB cable.

A computer running Windows, macOS, or Linux.

Install the Arduino IDE and configure the IDE as mentioned in the previous chapter

- Connect to the Internet and upload code 1 to find out the private IP so that it is placed in the Python file

Device 2

Install pynput library in Python

Install Python 3.6 or later.

Place the code in a new Python file (such as keylogger.py) specifying the protocol, port, and address of the esp32 because it will receive the results of this connection.

Steps:

Open the Arduino IDE.

Put code 2 in the new Arduino IDE window:

Select the appropriate board type and port (com4) for the ESP32 module from the Tools menu in the Arduino IDE, then upload the code to the ESP32 module while continuing to press boot until the upload is complete.

Open Serial Monitor and press rst until confirmation of connection to the network appears and confirm connection with the other device in the Arduino IDE to monitor communications and data received on the ESP32 device.

3.3.3 Possible problems in the project

Possible problems in the project of connecting ESP32 to Keylogger and how to solve them

1. Not connecting to Wi-Fi.

Possible reasons: (SSID or password error, Weak or no Wi-Fi signal, Problem with ESP32 module.)

Possible solutions:

Verify that the SSID and password in the code are correct.

Make sure the ESP32 is within Wi-Fi range.

Try restarting the ESP32 module. **P a g e | 30**

Use a network scanning tool to make sure your network is working properly.

2. Not receiving data on ESP32

Possible reasons: (Error in the IP address or port specified in the code, Network connection problem between computer and ESP32, Problem with Python code or ESP32 code.)

Possible solutions:

Make sure that the IP address and port in the Python code are the same as in the ESP32 code.

Verify that both devices are connected to the same Wi-Fi network.

Use network tools (such as ping or netstat) to check the connection between the computer and the ESP32.

Review the code to make sure there are no programming errors.

3. Interruption of communication between Keylogger and ESP32

Possible reasons: (Wi-Fi network outage, Problems with the performance of the ESP32 module or computer.)

Possible solutions:

Make sure your Wi-Fi connection is stable.

Try reducing the load on the network (for example, turning off other devices connected to the network).

Check the performance of the ESP32 and computer and make sure there are no performance issues.

4. Delay or loss in data transmission

Possible reasons: (Slow Wi-Fi network, Network congestion.)

Possible solutions:

Try reducing the distance between the ESP32 and the router to improve the Wi-Fi signal.

Make sure there is no network congestion by reducing the number of connected devices.

5. Problems running Keylogger

Possible reasons: (Error installing pynput library, Programming errors in Python code.)

Possible solutions:

Make sure the pynput library is installed using the `pip install pynput` command.

Verify the correctness of the code and correct any errors.

Try running the code on another device to verify there are no system issues.

6. Processing errors on ESP32

Possible reasons: (Programming errors in ESP32 code, Failure to properly handle received data.)

Possible solutions:

Make sure that the ESP32 code is written correctly and that there are no programming errors.

Add debugging messages in the code to print the received data and see how it is processed.

7. Battery drains quickly on ESP32

Possible reasons:

Continuous operation of ESP32 without power saving mode.

Use of additional components that consume high energy.

Possible solutions:

Enable power saving mode on the ESP32 when continuous operation is not required.

Use external power sources if consumption is high.

3.3.4 Detecting from keylogger

3.2.6 Using KeyScrambler

KeyScrambler for OS-level keystroke protection:

Installing KeyScrambler on Device 2:

You can download and install KeyScrambler from the official KeyScrambler website.

KeyScrambler configuration:

After installation, KeyScrambler automatically runs in the background to encrypt keystrokes at the system level.

Adjust settings if necessary to ensure target applications are protected.

Integrate KeyScrambler with Keylogger:

KeyScrambler works transparently with all applications, so you don't need to significantly modify the Keylogger code.

Make sure KeyScrambler is working and check that keystrokes are properly encrypted before sending them.

Since it is often used for malicious purposes, this calls for you to find ways to protect yourself from keyloggers and their intentions to ruin your computing experience and to protect your data. If a keylogger is installed on your computer, it will record all the keystrokes you press on your keyboard and may send them to a remote user. This means that a remote user can actually access your computer along with all the credentials/passwords and other information that is important to you.

-keep your system up to date with Windows Updates to keep software vulnerabilities to a minimum.

Being proactive about system security is always a good idea. One of the most important parts of proactive defense is keeping your system up to date and this includes your operating system as well as the applications and programs you run on it. Keyloggers and other malware look for exploits in outdated software and can take advantage of them, sometimes without you even knowing something is wrong.

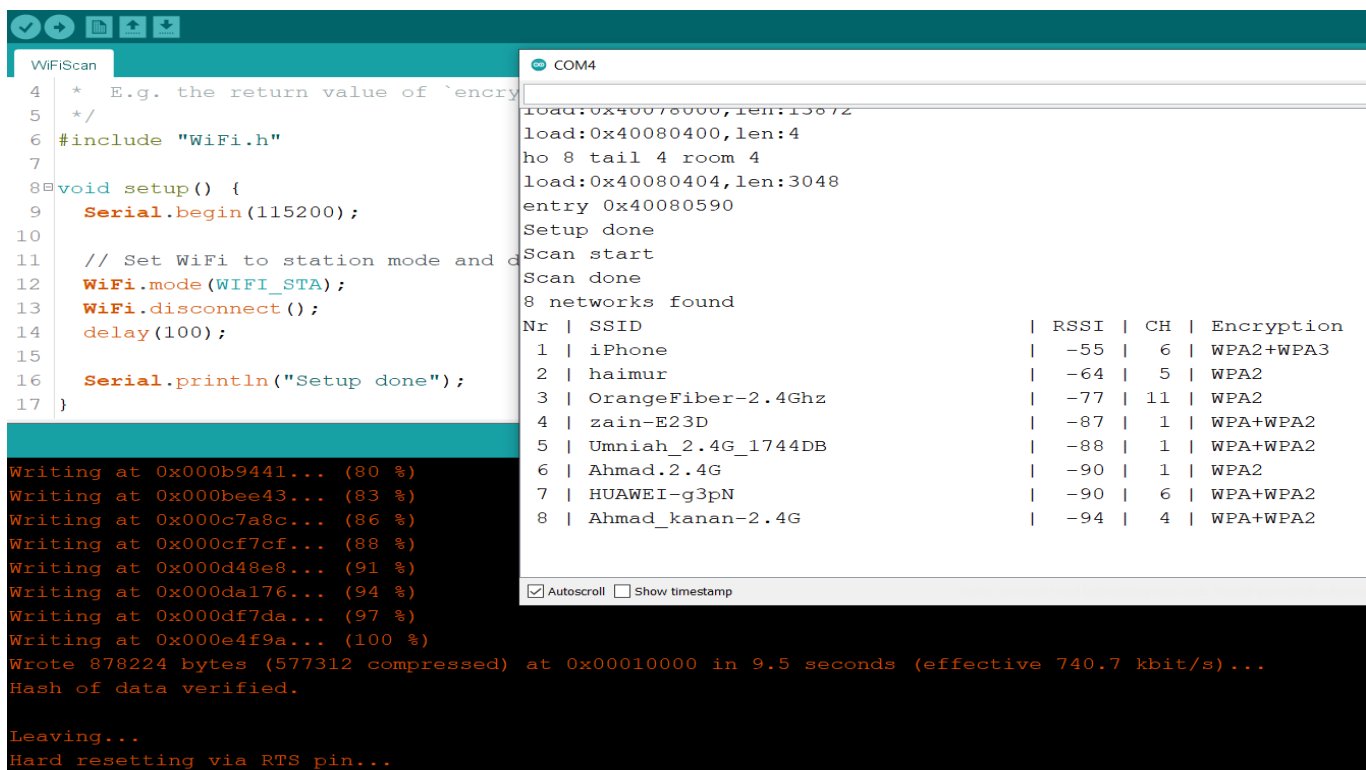
4.experiments

4.1Performance testing and evaluation

An explanation of the testing methods used to evaluate project performance.

In this project, the ESP32 was used to apply different testing methods to evaluate the project performance, which includes the following tests: Scanning available network from examples in IDE.

*Asking for IP address (private IP) and we can know public IP and scanning for networks



```
4  * E.g. the return value of `encry
5  */
6  #include "WiFi.h"
7
8  void setup() {
9      Serial.begin(115200);
10
11     // Set WiFi to station mode and
12     WiFi.mode(WIFI_STA);
13     WiFi.disconnect();
14     delay(100);
15
16     Serial.println("Setup done");
17 }
```

```
load:0x40078000,len:13872
load:0x40080400,len:4
ho 8 tail 4 room 4
load:0x40080404,len:3048
entry 0x40080590
Setup done
Scan start
Scan done
8 networks found
Nr | SSID | RSSI | CH | Encryption
1 | iPhone | -55 | 6 | WPA2+WPA3
2 | haimur | -64 | 5 | WPA2
3 | OrangeFiber-2.4Ghz | -77 | 11 | WPA2
4 | zain-E23D | -87 | 1 | WPA+WPA2
5 | Umniah_2.4G_1744DB | -88 | 1 | WPA+WPA2
6 | Ahmad.2.4G | -90 | 1 | WPA2
7 | HUAWEI-g3pN | -90 | 6 | WPA+WPA2
8 | Ahmad_kanan-2.4G | -94 | 4 | WPA+WPA2
```

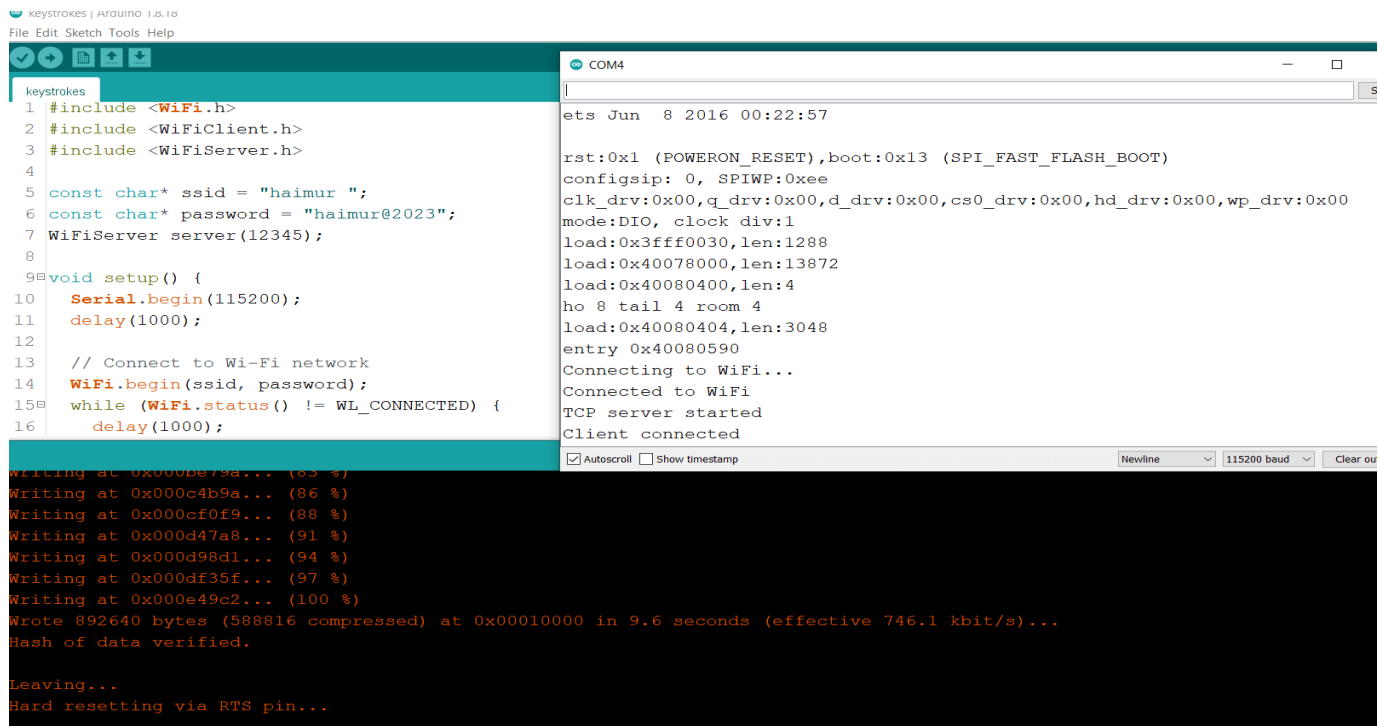
```
Writing at 0x000b9441... (80 %)
Writing at 0x000bee43... (83 %)
Writing at 0x000c7a8c... (86 %)
Writing at 0x000cf7cf... (88 %)
Writing at 0x000d48e8... (91 %)
Writing at 0x000da176... (94 %)
Writing at 0x000df7da... (97 %)
Writing at 0x000e4f9a... (100 %)
Wrote 878224 bytes (577312 compressed) at 0x00010000 in 9.5 seconds (effective 740.7 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
```

4.1 Performance testing and evaluation

4.experiments

We can now connect our keylogger with the ESP32, by adding ip address to our code in keylogger.

We can see that we are connect to the network and the client or taget also connected by and



```
keystrokes | Arduino 1.8.15
File Edit Sketch Tools Help

keystrokes
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <WiFiServer.h>
4
5 const char* ssid = "haimur ";
6 const char* password = "haimur@2023";
7 WiFiServer server(12345);
8
9 void setup() {
10   Serial.begin(115200);
11   delay(1000);
12
13   // Connect to Wi-Fi network
14   WiFi.begin(ssid, password);
15   while (WiFi.status() != WL_CONNECTED) {
16     delay(1000);
17   }
18 }
19
20 void loop() {
21   // Send data to server
22   if (WiFiClient.available()) {
23     String data = WiFiClient.readString();
24     server.write(data);
25     server.flush();
26   }
27 }

Writing at 0x00000000... (83 %)
Writing at 0x00000001... (86 %)
Writing at 0x00000002... (88 %)
Writing at 0x00000003... (91 %)
Writing at 0x00000004... (94 %)
Writing at 0x00000005... (97 %)
Writing at 0x00000006... (100 %)
Wrote 892640 bytes (588816 compressed) at 0x00010000 in 9.6 seconds (effective 746.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1288
load:0x40078000,len:13872
load:0x40080400,len:4
ho 8 tail 4 room 4
load:0x40080404,len:3048
entry 0x40080590
Connecting to WiFi...
Connected to WiFi
TCP server started
Client connected
```

we have to wait until upload to make sure we have no errors

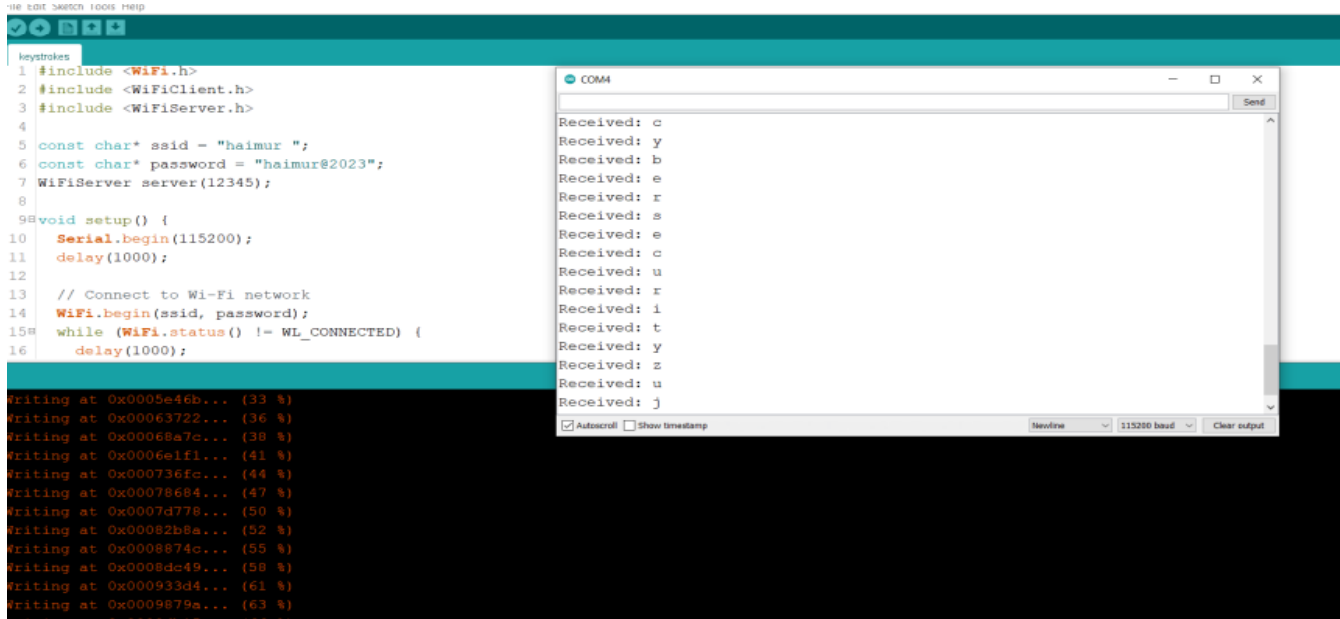
```
Writing at 0x00000000... (83 %)
Writing at 0x00000001... (86 %)
Writing at 0x00000002... (88 %)
Writing at 0x00000003... (91 %)
Writing at 0x00000004... (94 %)
Writing at 0x00000005... (97 %)
Writing at 0x00000006... (100 %)
Wrote 892640 bytes (588816 compressed) at 0x00010000 in 9.6 seconds (effective 746.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

4.1 Performance testing and evaluation

4.experiments

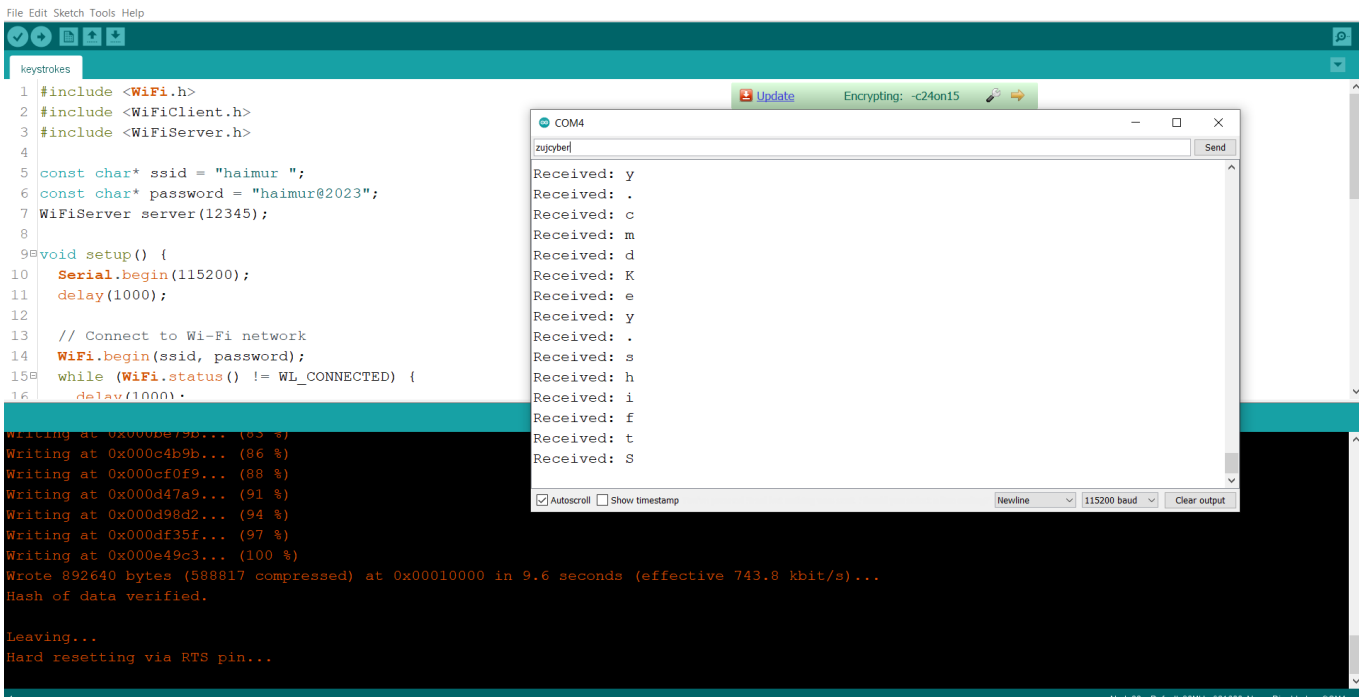
We can receive keystroke on our serial monitor.



The screenshot shows the Arduino IDE with a sketch named 'keystrokes'. The sketch includes `<WiFi.h>`, `<WiFiClient.h>`, and `<WiFiServer.h>`. It defines a server on port 12345 and a `setup()` function that initializes the serial port at 115200 baud and connects to a Wi-Fi network. The serial monitor shows the following received characters: c, y, b, e, r, s, e, c, u, r, i, t, y, z, u. The sketch also displays memory usage statistics at the bottom of the serial output.

```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <WiFiServer.h>
4
5 const char* ssid = "haimur ";
6 const char* password = "haimur@2023";
7 WiFiServer server(12345);
8
9 void setup() {
10   Serial.begin(115200);
11   delay(1000);
12
13   // Connect to Wi-Fi network
14   WiFi.begin(ssid, password);
15   while (WiFi.status() != WL_CONNECTED) {
16     delay(1000);
17   }
18
19   Writing at 0x0005e4eb... (33 %)
20   Writing at 0x00063722... (36 %)
21   Writing at 0x00068a7c... (38 %)
22   Writing at 0x0006e1f1... (41 %)
23   Writing at 0x000736fc... (44 %)
24   Writing at 0x00078684... (47 %)
25   Writing at 0x0007d778... (50 %)
26   Writing at 0x00082b8a... (52 %)
27   Writing at 0x0008874c... (55 %)
28   Writing at 0x0008dc49... (58 %)
29   Writing at 0x000933d4... (61 %)
30   Writing at 0x0009879a... (63 %)
```

After encryption we can see the difference between first response and this:



The screenshot shows the same Arduino IDE sketch, but with the 'Encryption' checkbox checked in the serial monitor window. The serial monitor shows the following received characters: y, ., c, m, d, K, e, y, ., s, h, i, f, t, S. The sketch also displays memory usage statistics at the bottom of the serial output.

```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <WiFiServer.h>
4
5 const char* ssid = "haimur ";
6 const char* password = "haimur@2023";
7 WiFiServer server(12345);
8
9 void setup() {
10   Serial.begin(115200);
11   delay(1000);
12
13   // Connect to Wi-Fi network
14   WiFi.begin(ssid, password);
15   while (WiFi.status() != WL_CONNECTED) {
16     delay(1000);
17   }
18
19   Writing at 0x000de79b... (83 %)
20   Writing at 0x000c4b9b... (86 %)
21   Writing at 0x000cf0f9... (88 %)
22   Writing at 0x000d47a9... (91 %)
23   Writing at 0x000d98d2... (94 %)
24   Writing at 0x000df35f... (97 %)
25   Writing at 0x000e49c3... (100 %)
26
27   Wrote 892640 bytes (588817 compressed) at 0x00010000 in 9.6 seconds (effective 743.8 kbit/s)...
28   Hash of data verified.
29
30   Leaving...
31   Hard resetting via RTS pin...
```

4.1 Performance testing and evaluation

4.experiments

we can see the difference of results after encryption, and the program still working and encryption.



Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	Status	4% CPU	39% Memory	0% Disk	0% Network	0% GPU	GPU engine
> Python (32 bit) (3)		0%	13.5 MB	0 MB/s	0 Mbps	0%	
Windows Defender SmartScreen		0%	5.3 MB	0 MB/s	0 Mbps	0%	
> Skype		0%	1.7 MB	0 MB/s	0 Mbps	0%	
Microsoft Windows Search Filter...		0%	1.0 MB	0 MB/s	0 Mbps	0%	
> Search	⚙	0%	0 MB	0 MB/s	0 Mbps	0%	
> Settings	⚙	0%	0 MB	0 MB/s	0 Mbps	0%	
> Task Manager		0.5%	23.6 MB	0 MB/s	0 Mbps	0%	
Microsoft Windows Search Prot...		0%	1.2 MB	0 MB/s	0 Mbps	0%	
> wsappx		0%	2.0 MB	0 MB/s	0 Mbps	0%	
KeyScrambler		0%	1.6 MB	0 MB/s	0 Mbps	0%	
> Service Host: Group Policy Client		0%	1.2 MB	0 MB/s	0 Mbps	0%	
> Service Host: Data Sharing Servi...		0%	2.7 MB	0 MB/s	0 Mbps	0%	
System Settings Broker		0%	5.5 MB	0 MB/s	0 Mbps	0%	
User OOBE Broker		0%	1.4 MB	0 MB/s	0 Mbps	0%	

< >

⬆ Fewer details

End task

5. Conclusion

Future applications and expansion

-Discussion on how to use this project in different applications.

It is possible to develop the project and use many different libraries with it and add other devices to add a keylogger, as the scan32 in this case will be a connection point or connected to a router and through the ARP protocol, and according to what the traffic has been analyzed, you will be able to monitor in detail.

Testers aim to test the security of the system and discover potential vulnerabilities. By using the ESP32 as a monitoring device, testers can analyze network traffic and detect any security vulnerabilities.

Some potential uses of a network monitoring project in penetration testing include:

Detect unauthorized devices on the network.

Traffic monitoring and analysis to detect hacking attempts.

Discover connected parties and verify their identity and security.

Test analysis of sent and received data to detect any unauthorized activities.

A good example of the capabilities of the ESP32 is its use with an ATmega32U4 to extract keystrokes from devices on a specific network. Due to the nature of the ESP32 as a smart device and the

difficulty of controlling it, it is necessary to periodically scan the network and monitor its traffic.

It is necessary to track ARP distribution and monitor network activities to identify any suspicious activity that may be indicative of hacking attempts. In addition, data sent over the network must be encrypted and high security measures applied, such as providing a higher level of security than WPA, and applying advanced encryption technologies such as WPA3 when possible.

Emphasizing the importance of paying attention to securing networks, comprehensive security strategies that include continuous monitoring and regular security updates must be developed and implemented to address evolving cyber threats. Using ESP32 in collaboration with ATmega32U4 presents innovative possibilities for developing effective security systems and enhancing data and network security.

Conclusions

-Summarize key results and evaluate achievements.

Capstone Project Explore the compatibility between the ESP32, a powerful and versatile microcontroller, and Keylogger, the well-known tool for logging keystrokes. The study began by analyzing potential compatibility scenarios between the two, focusing on how ESP32 could be used as a means of protecting data from keyloggers.

An experimental model based on ESP32 has been designed and developed to analyze and process inputs from the keyboard, and encrypt them before sending them to the target system. Available software libraries were used to facilitate this work, and extensive testing was conducted to ensure the system was efficient and effective in protecting data.

Through this project, promising results have been achieved indicating that ESP32 can be used as an effective tool to protect data from keyloggers and other potential threats. It was also concluded that this approach opens new horizons for security applications, such as securing mobile devices and electronic payment applications.

emphasize the importance of directing attention towards developing innovative and effective security solutions, in order to ensure the safety of sensitive data and personal information in an era of increasing cyber threats. By using technologies such as

ESP32 and Keylogger as learning and research tools, the academic and industrial community can leverage the findings to enhance cybersecurity and contribute to building a safer and more reliable digital environment.

In addition, the project demonstrated the importance of continuous research and development in the field of information security, and the necessity of innovation in finding innovative technical solutions to meet the increasing security challenges in the modern digital age.



جامعة الزيتونة الاردنية

قسم الامن السيبراني

مشروع تخرج

عمان, الاردن

عنوان المشروع
تم التحضير والاعداد بواسطة
خالد جمال حيمور

تم الاشراف على المشروع
د. بلال حواشين

2023-2024