WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Attiny85 Password Bypass

[1]Naveen Kumar T, [2]Nikil V, [3]Sabarish S, [4]Ms. P. Charanya M.E.

[1,2,3]Final Year/ Cyber Security, Mahendra Engineering College, Namakkal, Email: nk4169895@gmail.com
[4]Assistant Professor / Cyber Security, Mahendra Engineering College, Mahendhirapuri, Mallasamudram, Namakkal,
Email: charanyap@mahendra.info

ABSTRACT

A "brute force attack" typically involves systematically attempting all possible combinations of passwords, PINs, or encryption keys to gain unauthorized access to a system. while this attack method is commonly associated with more powerful computing systems, this abstract explores the conceptual idea of a brute force attack on an ATTiny85 microcontroller, a resource-constrained 8-bit microcontroller often used in embedded applications.

The ATtiny85 is a compact and versatile 8-bit microcontroller developed by Microchip Technology, formerly Atmel. this microcontroller is a member of the AVR family, known for its low power consumption and cost-effectiveness.

Despite its small size, the ATtiny85 offers a range of features that make it suitable for a wide array of embedded applications

1. INTRODUCTION:

Attempting a brute force attack on an ATTiny85-based USB device like the Digistump Digispark is a highly specific scenario and generally not a common use case for these devices. however, it's important to note that the ATtiny85 itself does not have native USB capabilities.

It lacks a built-in USB peripheral, unlike some other microcontrollers like the ATmega32U4 or ATmega328P to implement USB communication with an ATtiny85, an external USB-to-serial bridge or a similar solution is typically used

The ATtiny85 is a popular microcontroller from Microchip Technology (formerly Atmel) known for its small size and low power consumption.

1.1 USB functionality to an ATtiny85:

1)Understanding ATtiny85:

The ATtiny85 is a microcontroller with features such as GPIO pins, analog-to-digital converters, timers, and more, but it lacks native USB hardware.

2)Selecting a USB-to-Serial Bridge:

Choose a USB-to-serial bridge IC (e.g., FT232RL, CH340G) to establish a communication link between the ATtiny85 and a USB port. This bridge translates data between the ATtiny85's serial communication (e.g., UART) and USB protocols.

3)Wiring and Connection:

Connect the necessary pins of the ATtiny85 to the USB-to-serial bridge IC, ensuring proper wiring and electrical connections.

4)Serial Communication:

Program the ATtiny85 to use a serial communication protocol (e.g., UART) to send and receive data to/from the USB-to-serial bridge IC

5)Testing and Debugging:

Connect the ATtiny85 to a USB host (e.g., a computer) and test the USB functionality.

Debug and refine your code as necessary.

6)Power Management:

Implement proper power management for your application. USB can be a source of power for your device, but you may also need a power supply if the USB host doesn't provide enough power.

1.2 Project Objectives:

When adding USB functionality to an ATtiny85, you should focus on several key objectives to ensure a successful implementation. Here are the primary areas of focus and their associated objectives:

1.2.1 Hardware Setup:

Objective: Properly configure the hardware connections for USB. Ensure that the D+ and D- lines are connected correctly to the USB connector. Implement any necessary signal conditioning, such as pull-up resistors and capacitors.

1.2.2 Software Development:

Objective: Write, debug, and optimize the software that handles USB communication on the ATtiny85. This involves the following sub-objectives:

1)USB Stack Integration: Successfully integrate the V-USB library or other USB software stack into your firmware.

2)USB Descriptors: Define and configure USB descriptors for your device, including device descriptors, configuration descriptors, and endpoint descriptors.

3)Data Handling: Implement data transfer and endpoint communication routines to send and receive data over USB.

4)Interrupt Handling: Manage USB-related interrupts and event-driven code.

5)State Machine: Implement a robust state machine to handle different USB states, such as device enumeration and data transfer.

1.2.3 Code Optimization:

Objective: Optimize your code to fit within the ATtiny85's limited resources (flash and RAM). This includes minimizing code size and optimizing data storage.

1.2.4 Compatibility and Compliance:

Objective: Ensure that your USB implementation is compatible with USB standards and regulations. This includes sub-objectives like:

1)Vendor ID Assignment: If applicable, obtain a USB Vendor ID for your device.

2)USB Compliance Testing: Verify that your device complies with USB standards by conducting compliance testing if required.

1.2.5 Testing and Debugging:

Objective: Rigorously test your USB implementation to identify and fix any issues. This includes the following sub-objectives:

1)Functional Testing: Ensure that your device functions correctly when connected to a USB host.

2)Error Handling: Implement robust error handling to gracefully recover from unexpected USB events or issues.

3)Debugging Tools: Use debugging tools and techniques (e.g., logic analyzers, USB protocol analyzers) to diagnose and resolve problems.

1.2.6 Power Management:

Objective: Implement power management features to efficiently manage power consumption and handle various power scenarios (e.g., USB-powered or self-powered operation).

1.2.7 Documentation:

Objective: Maintain detailed documentation that describes your USB implementation, including hardware connections, firmware code, and any specific configurations. This documentation is essential for future maintenance and collaboration.

1.2.8 Security:

Objective: Implement security measures to protect your USB device from potential threats, such as unauthorized access or data breaches.

1.2.9 User Experience (UX):

Objective: Consider the user experience, including indicator LEDs, user interfaces, and error reporting, to make your USB device user-friendly

Remember that adding USB functionality to an ATtiny85 is a challenging task, and the limited resources of this microcontroller may require careful planning and optimization. Consider using a microcontroller with native USB support if your project's requirements demand more resources and capabilities.

2. EXISITUNG SYSTEM

2.1 Limitation of existing system

2.1.1 Limited Resources:

The ATtiny85 has limited resources compared to larger microcontrollers. It has a small amount of flash memory (8KB), SRAM (512 bytes), and EEPROM (512 bytes). This limitation can restrict the complexity and size of the programs that can be written for it.

2.1.2 Limited I/O Pins:

The ATtiny85 offers a limited number of I/O pins (6 in total), which may constrain the number of devices or components that can be interfaced directly with the microcontroller in a project.

2.1.3 No Native USB Support:

Unlike some other microcontrollers, the ATtiny85 does not have native USB support. Achieving USB communication often requires additional components and a USB-to-serial bridge, adding complexity and cost to the system.

2.1.4 Limited Clock Speed:

The ATtiny85 has a maximum clock speed of 20 MHz. While sufficient for many applications, higher-speed microcontrollers are available for projects that require faster processing.

2.2 H/w Specification

1) Microcontroller Family: AVR (Advanced Virtual RISC)

2) Bit Width: 8-bit

3) CPU and Clock Speed:

4) CPU Type: RISC

5) Maximum Clock Speed: 20 MHz

6) Instruction Set: AVR Enhanced RISC

2.2.1 Memory:

1) Flash Memory: 8 KB (program memory for storing firmware)

2) SRAM (Static Random-Access Memory): 512 bytes

3)EEPROM (Electrically Erasable Programmable Read-Only Memory): 512 bytes

4) Memory Organi0zation: Harvard Architecture

2.3 S/w Specification

2.3.1 Programming Language:

C and C++: Most applications for ATtiny85 are written in C or C++. These languages are widely used for microcontroller programming due to their efficiency and close-to-hardware capabilities.

2.3.2 Development Environment:

Arduino IDE: The Arduino Integrated Development Environment (IDE) is a popular choice for programming ATtiny85. Users can utilize the Arduino platform with the appropriate ATtiny85 core files and plugins to program the microcontroller.

2.3.3 AVR-GCC Toolchain:

The AVR-GCC (GNU Compiler Collection) is a commonly used toolchain for compiling C and C++ code for AVR microcontrollers, including the ATtiny85. It provides the necessary tools to compile, link, and create firmware for the microcontroller.

3. METHODOLOGY

3.1 Project Planning and Requirements Gathering:

Define the project objectives, functionality, and scope.

Gather requirements to understand what the system should accomplish.

3.2 Architecture and System Design:

Design the overall system architecture, defining the role of ATtiny85 and it interaction with other components.

Determine input/output devices, communication interfaces, power requirements, and the logic flow of the program.

3.3 Hardware Setup:

Set up the hardware components, including ATtiny85 microcontroller, input/output devices (sensors, actuators), and power supply.

3.4 Software Development:

Write the program logic and firmware for the ATtiny85 in a suitable programming language (e.g., C, C++).

Use development environments like Arduino IDE or AVR-GCC toolchain for writing and compiling the code

3.5 User Experience (UX):

Consider the user experience by adding indicator LEDs, user interfaces, and error reporting to make your USB device user-friendly.

3.6 Security:

Implement security measures to protect your USB device from potential threats, such as unauthorized access or data breaches.

3.7 Usability and Reliability:

Ensure that your USB device is user-friendly and reliable, working consistently and as expected in various usage scenarios…

3.8 Custom Ports:

Users with specific port ranges of interest can define custom scanning parameters, such as specifying the range or a list of ports to scan via command-line arguments.

3.9 Service Identification:

Service identification, a critical aspect of network analysis, is seamlessly integrated into the command-line interface.

4.Command-Line Interface

To create a command-line interface (CLI) for an ATtiny85 microcontroller over USB, you'll need to use a programming language like C or C++ and a library that allows communication between the ATtiny85 and your computer. The ATtiny85 microcontroller itself does not have built-in USB hardware, so you would typically use a library like V-USB (software-based USB) to implement USB communication.

Here's a general outline of the steps to create a basic CLI for an ATtiny85 over USB:

4.1 Set up your development environment:

Install the AVR toolchain, which includes the AVR-GCC compiler and related tools.

Install a text editor or integrated development environment (IDE) of your choice.

4.2 Write the firmware for the ATtiny85:

Create a C or C++ program that handles USB communication using a library like V-USB. You can find example code and documentation on the V-USB website.

Implement the CLI functionality in your program, which may include parsing commands, executing actions, and sending responses over USB.

4.3 Compile your firmware:

Use the AVR-GCC compiler to compile your code for the ATtiny85. Make sure to specify the correct microcontroller model and settings.

4.4 Flash the firmware onto the ATtiny85:

Use an AVR programmer (e.g., USBasp, AVRISP mkII) to flash the compiled firmware onto the ATtiny85 microcontroller.

4.5 Install USB drivers (if necessary):

Depending on your operating system, you may need to install USB drivers to recognize the ATtiny85 as a USB device.

4.6 Develop a host-side application (optional):

To interact with the ATtiny85 over USB, you may need to create a host-side application on your computer. This application can send commands and receive responses from the ATtiny85. You can write this application in a language of your choice, such as Python, C#, or C++

4.7 Connect the ATtiny85 to your computer via USB.

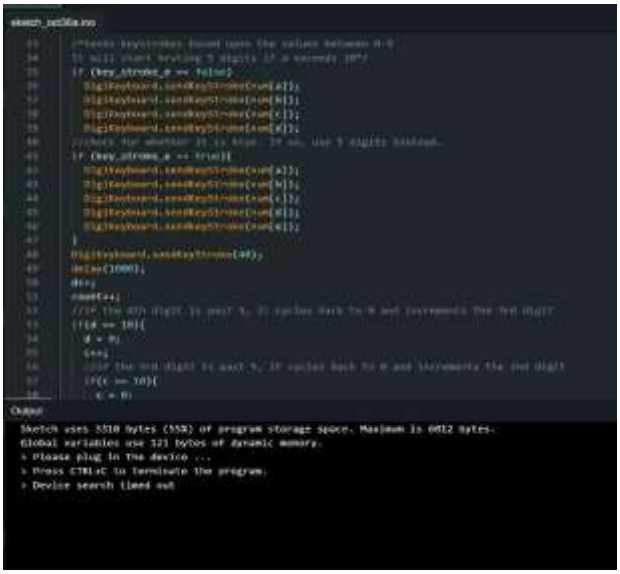
Run your host-side application to communicate with the ATtiny85 and test your CLI commands.

Remember that this is a high-level overview, and you'll need to refer to the documentation for the specific tools and libraries you choose to use. Additionally, working with USB on microcontrollers can be complex, so be prepared for some debugging and troubleshooting along the way..

4. RESULT:

Language –JAVA

Coding begin:



Output

Sketch uses 3310 bytes (55%) of program storage space. maximum is 6012 bytes.

Global variables use 121 bytes of dynamic memory.

>please plug in the device...

>press CTRL+C to terminate the program.

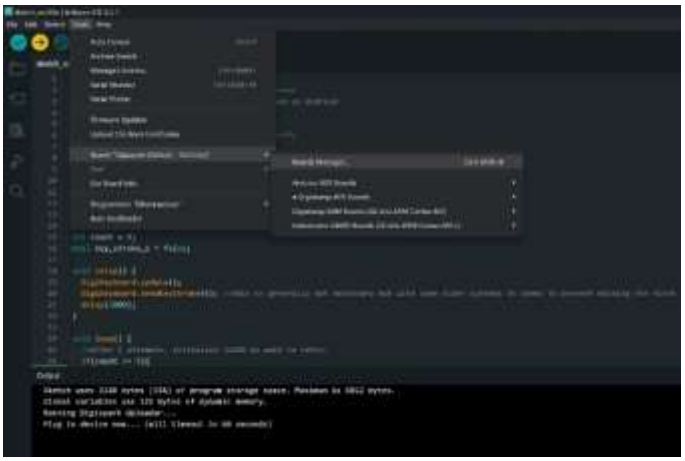
>Device search timed out

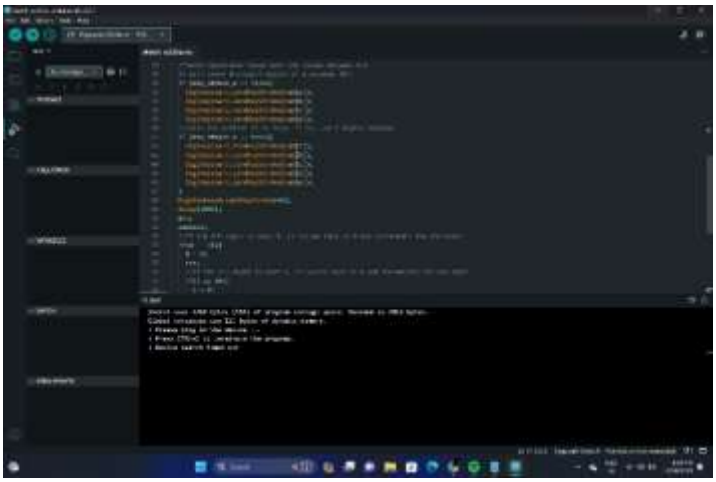
Output

Sketch uses 3310 bytes (55%) of program storage space. maximum is 6012 bytes.

Global variables use 121 bytes of dynamic memory.

>please plug in the device... (will timeout of dynamic memory)





Execution:

Real time experiment of attinty85

LIGHTS

>RED

>GREEN

GREEN LIGHT:

Green light I analyzing possible passwords

RED LIGHT:

Red light is final output showed.

After analyzing process is finding possible password mobile unlocked

Time limitations

10 sec - 1 to 10 bytes (numerical numbers)

60sec - 20 to 99 bytes (numerical numbers)

2 mins above - 100 to 200 bytes (numerical numbers)

Only 4 digits

5.DISCUSSION

5.1 Practical Applications:

Unlocking a mobile application using an ATtiny85 microcontroller could be applied in specific scenarios where the user intends to enhance the security or accessibility of a mobile app. This approach is subject to appropriate authorization and ethics, and it should be used responsibly

Two-Factor Authentication (2FA) Key:

The ATtiny85 can be programmed to act as a physical 2FA key for mobile applications. When the user attempts to log in, they would need to insert the ATtiny85 into their device to authenticate themselves. This can provide an additional layer of security beyond passwords or biometrics.

Secure Mobile Wallet:

If you have a mobile wallet app for managing cryptocurrencies or other sensitive financial assets, the ATtiny85 can be used to secure access to the wallet. It would only unlock the wallet when the physical ATtiny85 device is connected to the mobile device.

Enhanced Mobile App Security:

For apps containing highly sensitive or confidential information, the ATtiny85 can be used to secure access. This approach ensures that the user has the physical ATtiny85 device to unlock the app, adding a hardware-based security layer.

Parental Controls:

In a family setting, the ATtiny85 can be used to implement parental controls for certain mobile apps. Parents can provide the ATtiny85 to their children as a means to unlock specific apps, ensuring that children use them responsibly.

Corporate and Enterprise Apps:

For companies, particularly in industries with strict data security regulations, the ATtiny85 can be used to enhance security for mobile apps that access sensitive corporate data or perform specific business functions.

Privacy Protection:

Individuals who are concerned about privacy can use the ATtiny85 to lock specific apps that contain personal information, messages, or photos, ensuring that they remain protected from unauthorized access.

Mobile Password Manager:

The ATtiny85 can be used in conjunction with a mobile password manager app to add an extra layer of security. The user would need to connect the ATtiny85 to unlock and access their stored passwords.

It's important to emphasize that implementing this kind of solution should be done with the user's informed consent and with full respect for their privacy and legal rights. Any solution that aims to enhance security or accessibility should be developed and used responsibly, in compliance with relevant laws and ethical guidelines.

5.2 Implications:

Unlocking a mobile application using an ATtiny85 or any external device comes with several implications, both positive and negative, and it's crucial to be aware of these before considering its implementation.

Positive Implications:

Enhanced Security: By adding a hardware-based authentication factor like the ATtiny85, you can significantly enhance the security of the mobile application. It offers an extra layer of protection against unauthorized access, particularly useful for apps containing sensitive data or financial information.

Privacy and Control: Users can have more control over their data and privacy. They can choose to lock specific apps and ensure that their information remains confidential even if the device is physically accessible to others.

Two-Factor Authentication: It provides a convenient way to implement two-factor authentication (2FA). Users can combine their regular password or biometric authentication with a physical device for added security.

Parental Controls: For families with children using mobile devices, this approach can help parents implement parental controls more effectively, ensuring that children access only age-appropriate content.

Enhanced Security Posture: Accurate port scanning and service identification contribute to a more robust security posture, enabling organizations to proactively address potential threats.

Streamlined Network Management: The project simplifies device tracking and network optimization, resulting in more efficient network management.

5.3 STRENGTHS AND LIMITATIONS:

Strengths:

Enhanced Security: One of the primary strengths is the enhanced security it provides. The combination of traditional passwords or biometrics with a hardware-based authentication device like the ATtiny85 adds an extra layer of security, making it significantly more challenging for unauthorized users to access the mobile application.

Protection of Sensitive Data: It's particularly valuable for mobile applications that store sensitive data, such as financial information, health records, or confidential business data. Users can ensure the privacy and security of their data by requiring physical possession of the ATtiny85 to access the app.

Two-Factor Authentication (2FA): Implementing 2FA with a hardware token like the ATtiny85 is a robust security measure, making it difficult for malicious actors to gain unauthorized access to the app, even if they know the password.

Customization: Users have the flexibility to customize which apps or data they want to protect with the ATtiny85, allowing for tailored security measures based on their individual needs and preferences.

Parental Controls: It can be used to implement effective parental controls, ensuring that children or other users can only access approved applications.

Privacy and Control: Users have more control over their data and privacy. They can protect their information from prying eyes, even if the mobile device is unlocked 5.4 Future

Enhancements:

The TCP Port Capture Project lays a solid foundation for further development and improvement:

Protocol Expansion: Future versions could expand support to include UDP scanning and other protocols, broadening the tool's applicability.

Machine Learning Integration: Incorporating machine learning techniques may enhance service identification accuracy, particularly for identifying proprietary or non-standard services.

Scalability and Performance: Enhancements in scalability and performance can ensure the tool remains effective in large and complex network environments.

for malicious actors to gain unauthorized access to the app, even if they know the password.

Limitations:

Complex Setup: Implementing this security measure may require a degree of technical expertise. Users need to set up the ATtiny85 and configure the app to work with it. This can be a barrier to entry for some users.

Device Dependence: Users must have the ATtiny85 device with them at all times to access the mobile application. If they lose it or it becomes inoperative, they may have difficulty accessing the app.

Usability Concerns: The added security step can make the app more complex to use, potentially leading to user frustration and decreased usability. Striking a balance between security and user experience is crucial.

Cost: Acquiring the ATtiny85 device adds an extra cost for users, and it's their responsibility to manage it.

Compatibility Issues: Compatibility issues may arise, as the user's mobile device must support the ATtiny85 interface method (e.g., USB) and have the necessary drivers or software installed.

Data Recovery: In cases of device loss or damage, there must be a backup or recovery process in place to avoid permanent loss of access to the app.

Legal and Ethical Considerations: Implementing this system must be done within the boundaries of legal and ethical standards. Unauthorized access to a user's device or data could have legal repercussions.

Potential for Abuse: As with any security feature, there's potential for abuse. Users might misuse the system to control or lock someone else's device or apps without their consent

6.CONCLUSION

The ATtiny85 is a versatile microcontroller, appreciated for its small footprint, low power consumption, and cost-effectiveness. Its capabilities make it an excellent choice for a wide range of embedded applications, from consumer electronics to IoT devices, and it serves as a valuable tool in the field of microcontroller-based systems.

In conclusion, the concept of using an ATtiny85 to unlock a mobile application password offers both enhanced security and added complexity. This approach can be a valuable addition to mobile app security, particularly in situations where sensitive data or specific access controls are necessary. However, it should be implemented thoughtfully and responsibly, considering the following key points.

7. REFERENCES

Books:

[1] Attiny85 how to access a locked android phone via USB:

https://www.bing.com/ck/a?!&&p=5781e6c599d0df3bJmltdHM9MTY5OTIyODgwMCZpZ3VpZD0zMzZmMjZmMS0zNjhjLTYwNGMtMjdkYy0zNGYwMzcyYTYxZTUmaW5zaWQ9NTI2Ng&ptn=3&hsh=3&fclid=336f26f1-368c-604c-27dc-34f0372a61e5&psq=attiny85+references+books&u=a1aHR0cHM6Ly93d3cuZXRlY2hub3BoaWxlcy5jb20vYXR0aW55ODUtcGlub3V0LXNwZWNzLWd1aWRlLw&ntb=1

[2] GitHub:

https://www.bing.com/ck/a?!&&p=d876a3d87c68fbe9JmltdHM9MTY5OTIyODgwMCZpZ3VpZD0zMzZmMjZmMS0zNjhjLTYwNGMtMjdkYy0zNGYwMzcyYTYxZTUmaW5zaWQ9NTIyNw&ptn=3&hsh=3&fclid=336f26f1-368c-604c-27dc-

34f0372a61e5&psq=usb+mobile+application+password+bypass+books+reference&u=a1aHR0cHM6Ly93d3cubW9iaWtpbi5jb20vcGhvbmUtdW5sb2NrL2FjY2Vzcy1sb2NrZWQtYW5kcm9pZC1waG9uZS12aWEtdXNiLmh0bWw&ntb=1 Links

[3] ATTENY85 DATA SHEET : ATTINY85 Datasheet(PDF) - ATMEL Corporation (alldatasheet.com)

[4] ATTENY85 USB INTERFACE EXPL: Use ATtiny85 as USB interface? - Using Arduino / Microcontrollers - Arduino Forum

[5] Digispark-scripts . GitHub Topics: https://www.bing.com/ck/a?!&&p=c0a2d0324d434491JmltdHM9MTY5OTIyODgwMCZpZ3VpZD0zMzZmMjZmMS0zNjhjLTYwNGMtMjdkYy0zNGYwMzcyYTYxZTUmaW5zaWQ9NTIxNg&ptn=3&hsh=3&fclid=336f26f1-368c-604c-27dc-34f0372a61e5&psq=brute+force+attack+attiny85+usb+projects&u=a1aHR0cHM6Ly9naXRodWIuY29tL3RvcGljcy9kaWdpc3Bhcmstc2NyaXB0cw&ntb=1