Securing HTTP Communication for Homemade IoT Projects

Author: Jose R. de la Vega
Advisor: Dr. Jeffrey Duffany
Computer Science: IT Management and Information Assurance

Abstract

The Internet of Things (IoT) is a topic in the Computer Science and Engineering field that has rapidly grown in the last couple of years. The popularity of the topic has made different companies to make proprietary IoT devices which can be purchased with ease. Although these devices are very accessible now days, there are many students and professionals in the STEM area that like to create their own IoT devices using tools such as Raspberry Pi and Arduino. Working with these devices in such projects will definitely give students a boost in networking knowledge, but many miss the security part of the project. Using AES encryption to share messages and RSA public key encryption to encrypt the AES key can greatly improve the security of the HTTP communication for these projects as well as improving the student's knowledge in network security.

Introduction

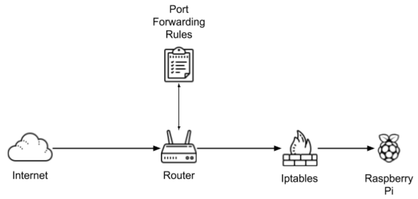
The Internet of Things (IoT) is a trending topic in the Computer Science and Engineering area. Creating your own IoT device using tools such as the Raspberry Pi can be very educational and inexpensive for students an STEM professionals. The Raspberry Pi offers great resources for people to develop all sots of projects, but people mainly focus on the functionality of the device they are building and not the security. The goal of this research is to provide a guide for people interested in making their home- made IoT devices on how to apply security to the system.

Materials

- **Raspberry Pi 3 Model B+**
- **Breadboard**
- **GPIO Pin Bridge to Breadboard**
- **220 ohm Resistors**
- **LED (red, blue, green)**
- **Jumper Wires**
- **Raspbian**
- **Python and PyCrypto**
- **React-JS and Crypto-JS**
- **Wireshark**

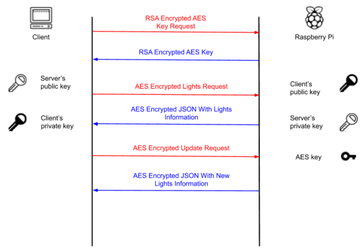
Methodology

1. **Setting Up the Raspberry Pi:**
   - Connect the Raspberry Pi to the network.
   - Give the Raspberry Pi a static IP address.
   - Crete rules for port forwarding in the router.
   - Create a firewall for the Raspberry Pi using Iptables for more security.
   - Assemble the circuit using the breadboard, LEDs, jumper wires, resistors, and GPIO bridge.

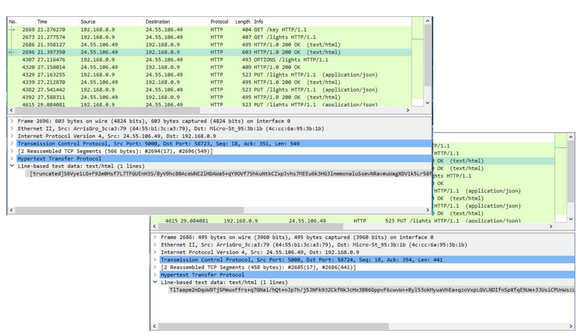


2. **Building the code for the client and server:**
   - **Client:**
     - Developed using React JS.
     - It sends a GET request to the server to get information about the lights.
     - When a button is pressed it sends a PUT request and a JSON to the server with the information of the changes:
       - GPIO pin that we want to change.
       - Status of the GPIO that we are changing.
   - **Server:**
     - Using Flask to develop the web server.
     - Using REST API to send and receive JSON data.
       - When a GET request for light status is received, the server sends a JSON with the lights.
       - When a PUT request for updating the lights is received, the server updates the json and turns on or off the lights.
     - Turn on/off the lights depending on the status received in the JSON using the GPIO pins.

3. **Applying security measures to the system:**
   - **AES:**
     - Used Advanced Encryption Standard (AES) to encrypt the communication between server and client.
     - Encryption is done by using PyCrypto in the server and Crypto-JS in the client.
     - AES is a fast encryption method that uses a symmetric key.
     - Symmetric encryption means that both the sender and receiver of a message need to have the same shared key.
     - Needs a secure method of sharing the key.
   - **RSA:**
     - Used RSA public key encryption to encrypt the AES key used in the communication.
     - This encryption method is slower than AES, which makes it not efficient to be used in all the communications.
     - It is an asymmetric encryption algorithm which means that both parties do not need the same key.
     - Both the sender and the receiver have their own set of keys; a public key and a private.
     - The public key is used to encrypt the message.
     - The private key is used to decrypt the message.
     - The key pairs where generated using PyCrypto.



4. **Sniffing the network traffic:**
   - There are three instances of the project where a sniffing test was performed using Wireshark:
     - Before adding any encryption: All JSON messages where seen in plain text, revealing information about the status of the lights of the circuit.
     - After adding AES encryption: The JSON messages where encrypted, and the information about the lights status was not readable, but the AES key that was shared between the client and the server was in plain text.
     - After adding AES and RSA encryption: Both the JSON message and the AES key where encrypted and no information could be seen by a hacker.



Conclusion

This project is a great way for students and STEM professionals interested in IoT to be able to develop a homemade IoT device while learning important concepts in network security. The project can be considered a homemade version of Open SSL. There are a lot of software designed for Raspberry Pi that serve as IoT controllers, but it takes away the users experience to learn about security. I made the project using inexpensive items so that students with limited resources could follow along and learn about network security for IoT devices. By using HTTP, the user avoids spending money on SSL certificates, and he/she gains knowledge on how the perform the encryption used in said technology. Overall, I'm very proud of the project and I think every student or STEM professional interested in learning about homemade IoT devices and HTTP communication security should give this project a try.

Future Work

The system is far from being completely secure. In this project I focused on the HTTP communication security. Although I did add other security measures to the Raspberry Pi, such as the Iptables firewall and disguising the ports used with port forwarding, the system needs other security measures. The first thing missing is authentication for the system. As of right now, anyone that has the client can turn on and off the LEDs without authenticating. Another thing listed as future work would be to create a more complex and useful circuit to better demonstrate the importance of applying encryption to the communication.

Acknowledgements

I want to thank my advisor, Dr. Duffany for all his support throughout the project. I also extend my gratitude to the National Science Foundation (NSF) and Cybercorps: Scholarship for Service (SFS).

References

1. Brix. (2019, January 4). brix/crypto-js. Retrieved from https://github.com/brix/crypto-js
2. Clark, J. (2017, September 19). What is the Internet of Things, and how does it work? Re- trieved from https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/
3. Flask. (n.d.). Retrieved from http://flask.pocoo.org/
4. Fritzing. (n.d.). Retrieved from http://fritzing.org/
5. Pycrypto. Retrieved from https://pypi.org/project/pycrypto/
6. Raspberry Pi 3 Model B – Raspberry Pi. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
7. SSL Certificate Digital Certificate Authority. (n.d.). Retrieved from https://www.ssl.com/
8. Wireshark. (n.d.). Retrieved from https://www.wireshark.org/
9. Icons8 Retrieved from https://icons8.com/icons