CMPE-361 Intro to Hardware Security

Project 2

Final Paper

Barak Binyamin
Performed December 12th, 2023
Submitted December 14th, 2023

Lab Section 01L1
Instructor: Dr. Michael Zuzak, Ph.D
TA: Long Lam, Colin Vo, Sydale John Ayi
Lecture Section 0
Professor: Dr. Michael Zuzak, Ph.D

Introduction

This exercise explored the hardware security of wireless communications. First, a device was created to emulate a common IOT device on a wireless network. Second, the device's network traffic was analyzed and an attack was devised to change the functionality of the device. Third, a patch was developed to prevent the attack from being successful in the future.

Theory

IOT devices are pieces of hardware, like gadgets or appliances, that communicate over the internet or other networks[1]. One common IOT device is a smart plug, an outlet that can be controlled from an app or website. These devices are very useful for home automation, gathering energy analytics, or industrial control [2][3]. Some visuals of smart plugs can be seen in Appendix 1.

In this exercise, a device to emulate the functionality of a smart plug was developed to consider possible strategies for attacks. This emulation device, the IOT-light, used WiFi to connect to the internet, check for updates and communications by connecting to a custom server, and if an update was available, would pull a new firmware from github.com, a free online code oriented storage and sharing platform.

There are many options to consider when connecting a device to the internet, but there are generally two categories; wired, and wireless. Many commercial smart plugs connect to the internet over WiFi, a suite of wireless protocols commonly used for local area networking. Wired connections have the drawback of requiring a physical tether to a device, limiting mobility, while wireless connections offer a large range of mobility, and often make connecting to the internet from location to location seamless [4], but there are also security risks to consider. Following the STRIDE methodology, it is crucial to identify assets, possible adversaries, features that enhance or weaken a design, as well as some threats. It is also crucial to identify the severity of each vulnerability, this can be done using the DREAD+D approach[5]. Assets include functionality of the device, usage statistics, and sensitive information embedded or passing through the air (wireless communication). Adversaries include saboteurs of an industrial or small-scale nature within range with access to a WiFi capable device. Wireless devices have the ability to receive wireless communication within range of transmission. Most wireless devices also have the ability to transmit information on the same frequencies. A device with wireless capabilities may have the ability to access communications, interrupt communications, tamper with communications, spoof authorship of communication, and even change device functionality as a side effect of alternating communications, possibly elevating the privileges of that wireless device. The threats are all inclusive, spoofing, tampering, information disclosure, repudiation, denial of service, & elevation of privilege while the respective vulnerabilities are the authentication, integrity of the device, confidentiality of the data being transmitted, non-repudiation, availability of functionality, and authorization of use. Assume the DREAD+D (Damage, Reproducibility, Exploitability, Affected Users, Discoverability, Detection) categories were each graded on a 1-5 scale, 1 being low, 5 being high severity. The damage from an unfunctional device can be quite severe, especially in mission critical environments, (ex. Bank servers powered through a smart plug), for that reason the threat of a denial of service attack should be rated high, 5/5. A wireless device capable of attacks can cost $5 or less, so the reproducibility category should be rated 5/5. Every user dependent on a service powered through a smart plug would be affected by this attack, so the Affected Users category should earn a 5/5. Wireless signals are often easy to discover using a $20 software defined radio and free software like gnuradio[6], or wireshark[7] so discoverability would earn a 2/5. Detection can be difficult, depending on how subtle the attack is as attacks can range from denial of service which is very noticeable, while tampering with communications can be less noticeable earning this category 3/5. That concludes the categories of DREAD+D, making the threat level of this analysis 20/25, a HIGH threat level. The hardware decision of exposing communications to the air produces more risk. When physical access to a communication medium is extended, the attack surface is physically larger. It is crucial to identify inexpensive and effective methods of defending against malicious wireless devices, as their impact may be unrecoverable.

For background, some key concepts for how smart phones, laptops, and IOT-devices interact with the internet are discussed in Appendix 2a & 2b.

To launch a comprehensive attack on the IOT-light it is crucial to first gain an understanding for how it is receiving commands to turn on and off the light, and how it is receiving firmware updates. Given that the IOT-light is built to connect to the internet over WiFi, network traffic can be redirected to a computer for analysis using a custom router. From there tools like wireshark, tcpdump, and some custom servers could be used to view the network traffic. Specifically, the domain names, HTTP(S) URL's, and raw data could be captured.

For the IOT-light to connect to the malicious router, the original connection must be broken. An open source WiFi attack and analysis tool called the ESP32 Marauder is capable of flooding the air with deauthentication messages tailored so they will disconnect devices from a specific router [8]. A diagram of this process can be seen in Figure 1.

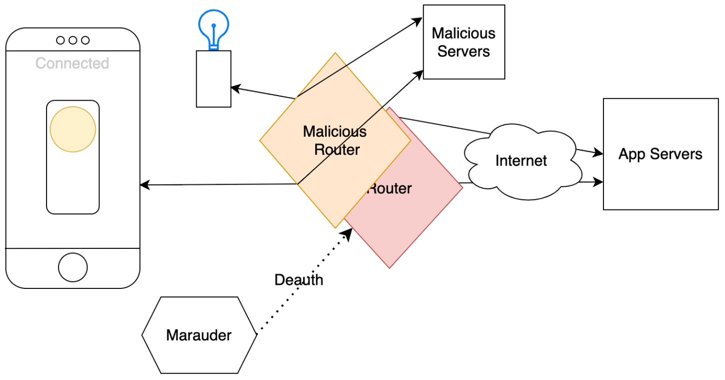


Figure 1: Deauthentication Attack on IOT-light and Router

Disconnecting devices often try to reconnect to a WiFi with the same credentials, to take advantage of that, a router with the same SSID and Password can be launched so the IOT-light will likely connect. After that, custom servers could be launched to imitate the behavior of the real IOT-light servers, change control logic, signal the IOT-light that an update is needed, and inject malicious firmware to gain full access.

Finally, to mitigate this attack, a verification of authorship is necessary. This can be accomplished by verifying the signature of the TLS certificate returned upon each HTTPS request. The IOT-lights firmware was already pulled using HTTPS (from https://github.com), but the signature was not verified, this requires additional steps. A diagram of a TLS certificate can be seen in figure 2.

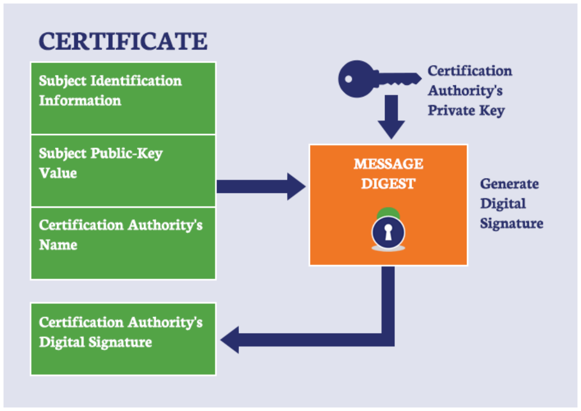

CERTIFICATE
Subject Identification Information
Subject Public-Key Value
Certification Authority's Name
Certification Authority's Digital Signature
Certification Authority's Private Key
MESSAGE DIGEST
Generate Digital Signature

Figure 1: TLS certificate taken from thesslstore.com

Updated browsers including Chrome and Safari, verify the authorship of a website's certificate through the verification of its digital signature. An https website can either have a self signed certificate or a certificate signed by a certificate authority. Common browsers and advanced IOT devices have a list of various certificate authorities and their public keys, which can be used to decrypt a portion of a certificate with their signature. Only a certificate organization's private key can produce a signature that will be successfully unencrypted by a public key. In this way, the signature is verified, and the connections can be evaluated as secure or insecure.

Measurement Methods

This exercise utilized multiple ESP-WROOM-32 development boards. These boards are inexpensive WiFi and bluetooth enabled devices capable of being programmed using platformIO and the Arduino framework[9]. One device was used as an ESP32 Marauder WiFi analysis and attack tool. Another was used as a simple router to redirect dns traffic to the first device connected. The third was used to make the target IOT-light, for which serial messages and LED's were used to verify the state of the device. A reference to the code & documentation to create all three devices is available in Appendix 6.

Results & Analysis

An IOT-light and custom servers were created to emulate an IOT-device. It was built with credentials to connect to a WiFi call "NotWifi", then check for updates and communications by connecting to a custom server, and if an update was available, would pull a new firmware from github.com. Examples of the device functioning as expected(on, off, no firmware update, firmware update) can be seen in Appendix 3.

The next step was analyzing wifi traffic to create an attack. The light was disconnected from the wifi, reconnected to a malicious router, which redirected all network traffic to a computer hosting an http analysis server that logged all domain names and URLs requested from the light. The first version of the analyze server found that websockets and "/version" were being requested on the domain mbinyamtorsMBP2.rochester.rr.com. The second version of the analyze server returned a high number

(22) on the request for "/version" to see if that would change the behavior of the requests. This approach worked in getting the light to request "/whatfirmwareurl" on the same domain. The results of this process can be seen in Appendix 4a. The next step was to try to launch an attack to get the IOT-light to a new malicious firmware by providing a URL.

An example of a successful attack in which the IOT-light was disconnected using the ESP32 marauder, reconnected to a malicious router, and directed to download malicious firmware can be seen in Appendix 4b.

Appendix 5 shows an attack on the patched version of the IOT-light that fails to direct the IOT-light to download the malicious firmware.

Discussion and Conclusions

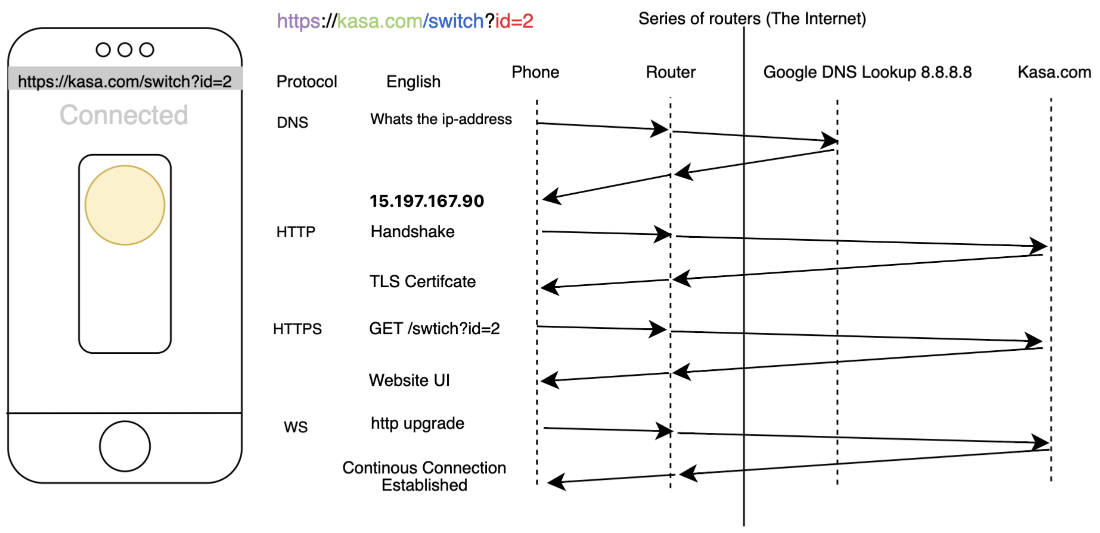
This exercise explored the hardware security of wireless communications, specifically WiFi. To do this, an IOT-light was created to emulate a smart plug on a wireless network. The IOT-light's network traffic was then analyzed and an attack was devised to change the functionality of the device. Lastly, a patch was developed to prevent the attack from being successful in the future. Verification of authorship was found to be the vulnerability. The scope of the problem of authorship extends beyond WiFi into any communications. Signed TLS certificates solve this problem by providing a way to verify if a new communication channel can be trusted.

Appendix

Appendix 1: Smart Plugs From Kasa & WYZE

Images taken from kasasmart.com & wyze.com

Appendix 2a: Overview of a Device Connecting to a Website



In this theoretical example, a phone is connected to kasa.com to control a smart plug. First the phone must learn the ip-address(like a phone number for the internet) of kasa.com by making a DNS(domain name service request). Once the request is resolved(address is received) the phone can request to connect. Kasa.com routes all http requests to https and sends back a public certificate to encrypt data. What's not shown in the image is some of the middle steps where the iphone sends back its public key so the server can also encrypt communications. Both the iphone and kasa.com have a private key that can be used to decrypt communications encrypted with their public key. This encryption scheme is in the category of Asymmetric. Finally, the website's files are sent to the phone, which contain code that the browser will use to display the UI, and upgrade the connection to websockets, a continuous communication protocol used for streaming event information between clients and servers.

The phone is connected to the router over WiFi*, a protocol that uses an SSID and Password (Network name and Encryption Key Base) to establish communications between devices like a router and phone so the router can arbitrate requests to communicate with devices locally and on the internet*

Keywords: ssid, password, ip-address, domain-name, DNS domain name service, http(s), websockets

Appendix 2b: Parts of a URL

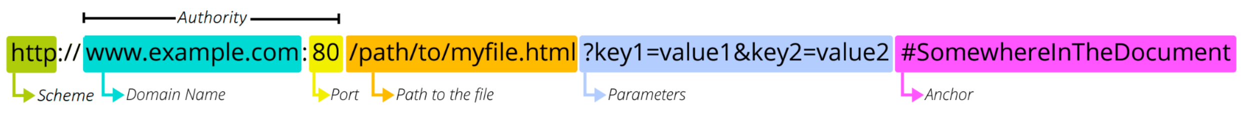
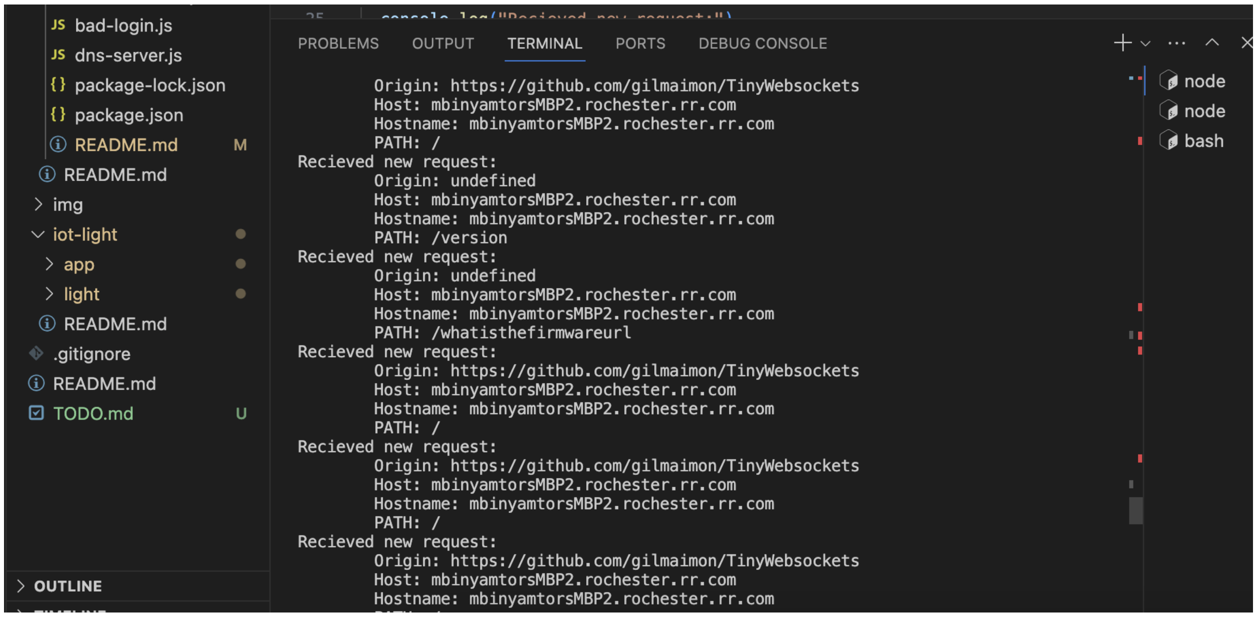

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Image Taken from [developer.mozilla.org]()

Appendix 3: Expected Functionality of the IOT-light

The video referenced shows the IOT-light toggled on and off, a boot with no firmware update available, and a boot with a firmware update (debug messages over serial @baud rate 115200)

Appendix 4a: Attack on IOT-light - HTTP & HTTPS Analysis Server & Results



```
        Origin: https://github.com/gilmaimon/TinyWebsockets
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
        PATH: /
Recieved new request:
        Origin: undefined
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
        PATH: /version
Recieved new request:
        Origin: undefined
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
        PATH: /whatisthefirmwareurl
Recieved new request:
        Origin: https://github.com/gilmaimon/TinyWebsockets
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
        PATH: /
Recieved new request:
        Origin: https://github.com/gilmaimon/TinyWebsockets
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
        PATH: /
Recieved new request:
        Origin: https://github.com/gilmaimon/TinyWebsockets
        Host: mbinyamtorsMBP2.rochester.rr.com
        Hostname: mbinyamtorsMBP2.rochester.rr.com
```

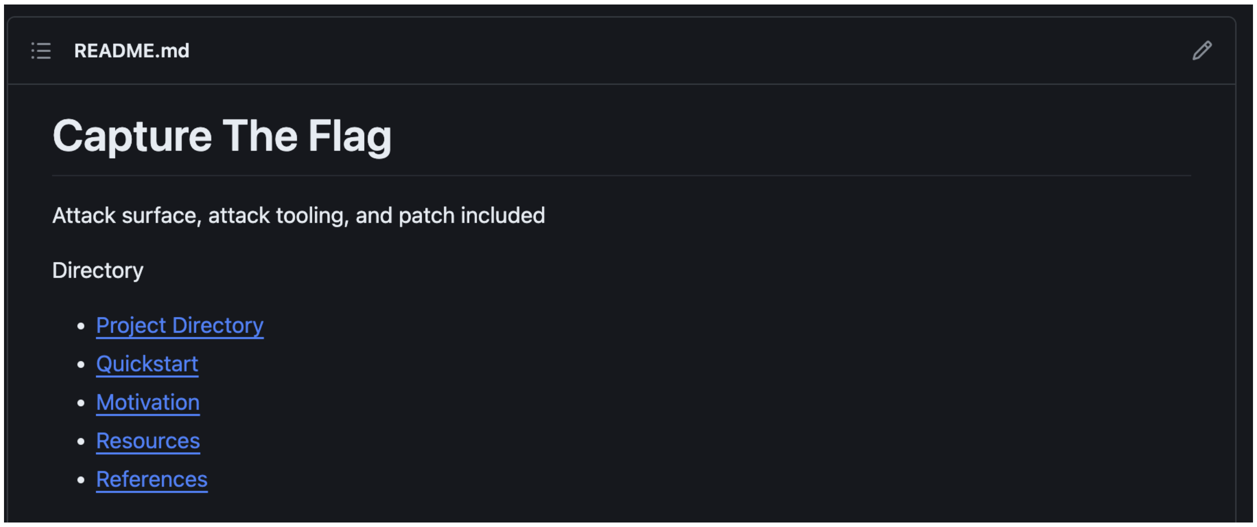
Appendix 4b: Attack on IOT-light - malicious firmware

The video referenced shows the IOT-light connected to the normal wifi, booted off, then reconnected to a malicious router which redirects traffic to malicious server which supplies the IOT-light with malicious firmware

Appendix 5: Attack on Patched IOT-light

The video referenced shows the IOT-light connected to the normal wifi, booted off, and then reconnected to a malicious router which redirects traffic to malicious server which supplies the IOT-light with malicious firmware, but the IOT-light does not accept it

Appendix 6: Code & Documentation to Create an IOT-light & Attack Tooling


README.md

Capture The Flag

Attack surface, attack tooling, and patch included

Directory

- Project Directory
- Quickstart
- Motivation
- Resources
- References

The git repository is listed at https://github.com/BarakBinyamin/ctf

References

[1] arm, what are IOT-devices, https://www.arm.com/glossary/iot-devices
[2] Wyze, *Wyze Smart Plugs*, https://www.wyze.com/products/wyze-plug
[3] Kasa, *Kasa Smart Plugs*, https://www.kasasmart.com/us/products/smart-plugs
[4] Cisco, *What Is Wi-Fi?* , https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html
[5] Dr. Michael Zuzak, *CMPE361 Threat Modeling Lecture Notes*, Contact For Access
[6] GNUradio, GNUradio, https://www.gnuradio.org/
[7] Wireshark, Wireshark, https://www.wireshark.org/
[8] justcallmekoko, *Deauth Attack*, https://github.com/justcallmekoko/ESP32Marauder/wiki/attack
[9] Rui & Sarah Santos, *Getting started with the ESP32*, https://randomnerdtutorials.com/vs-code-platformio-ide-esp32-esp8266-arduino/