Abertay University

Portable NMAP Scanner

Declan Doyle
1600219

CMP408: Cyber Security and System Internals

BSc Ethical Hacking
Year 4

2019/2020

1. Introduction

NMap is a network mapping tool which scans devices on a network for open ports, services, and performs software version detection. (Lyon, 1997) It is an essential tool for security vulnerability assessments and any individual looking to protect their network. This project aimed to create a device capable of running Nmap within a network that could be initiated by a website, and showed the status of a scan by LEDs on the device. This device should ensure that any network can be protected, as results can be viewed from anywhere, and a scan can be activated remotely if needed. The device can be used to provide reports on a home network, or connected to a network that requires a vulnerability test, allowing it to be used by both consumer and security professionals.

1.1. Objectives

- Make the Raspberry Pi run NMAP with settings passed to it
- Change the GPIO pins on the Pi to either show standby, scanning, or finished
- Pass commands from an EC2 instance to the Raspberry Pi
- Display results from scan on EC2 instance

2. Procedure

2.1. Overview and Operation

The solution is made of of 3 key parts: the bash scripts that control the operation of the device, the Raspberry Pi driver which operates the GPIO pins to manipulate the LEDs and buzzer alongside an application to control the driver, and an AWS EC2 instance to host a web server and MQTT broker to allow for input and output from the user, and to communicate with the Raspberry Pi. The operation of the device involves all 3 parts. Firstly, the user should enter information into the web application. This must be an IP address, a selection of the type of scan to run, and to chose whether the scan should output verbose results. Once the user selects submit, the web application will write a script file with the appropriate NMap command that corresponds to the options the user has entered. Following this, the web application runs the first script, which copies the script file to the home directory of the EC2 instance. This script also launches a python file which publishes to an MQTT topic. Meanwhile, on the Pi, upon setup of the device, a python script is run which sets the Pi in a listening state, subscribed to the same MQTT topic. Once the topic is published to with the appropriate message, the second script is launched, which downloads the command script file from the EC2 instance using SCP. This then launches the main script. This script makes the command file executable, and disengages the amber LED, indicating to the user that a scan is about to start. The main script file then executes the command file, and whilst the NMap scan is running, uses the Pi Driver and application to toggle the red LED, indicating a scan is taking place. Once the scan is completed, a command called Xsltproc (Veillard, 1999) is used to convert the xml output from the NMap scan to html. This is then copied to the EC2 instance using SCP, and then a script file using SSH is executed to copy this file from the EC2 home directory to the directory where the html files are located for the Apache server. The Pi driver and application will then sound a buzzer for 5 seconds to indicate the scan has completed and the results are ready to be viewed. The amber LED will then be reengaged signifying the device has returned to standby mode. The python script will be ran again to set the Pi to listening mode. On the web application, clicking the link to view the results will take the user to the NMap results that were just copied to the server. Using the browser's back button, the main page can again be viewed, and the process can be repeated. For a graphic representation of the above, see figure 1.

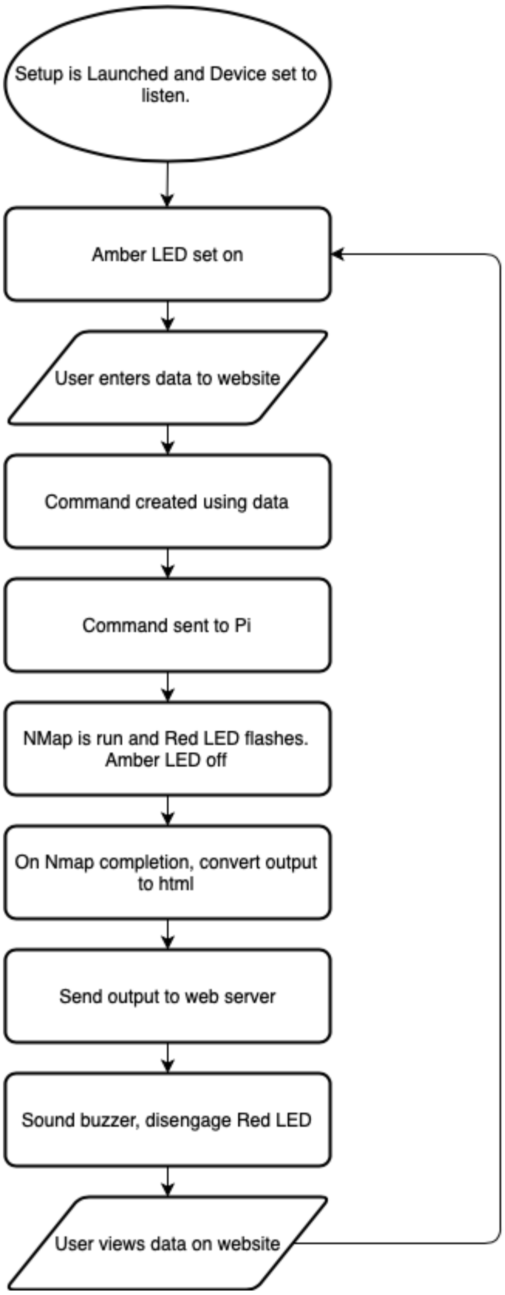
Setup is Launched and Device set to listen.
Amber LED set on
User enters data to website
Command created using data
Command sent to Pi
NMap is run and Red LED flashes. Amber LED off
On Nmap completion, convert output to html
Send output to web server
Sound buzzer, disengage Red LED
User views data on website

FIGURE 1

2.2. Scripts

There are 6 scripts in total in this projects, 4 are bash scripts, and the remaining 2 are python scripts. The first is not used by the solution directly, but is instead a bash script used to setup the Pi to be used without direct user input. The script will insert the driver module, turn on the standby light, and start the next script, which is the python MQTT subscribe script. This connects to the MQTT broker with a set username and password, and uses a certificate to enable encryption. The next script is run when the user submits the web form on the web server. This script copies the command file to the home directory as the html directory cannot be accessed without elevated privileges, so this allows for the command file to be transferred to the Pi without giving the Pi root privileges on the EC2 instance. This script then calls the next script, which is the MQTT publish python script, which publishes to the subscribed topic, signalling to the Pi to call the next script, which will download the command file and begin the scan process detailed above. The use of an MQTT broker to signal the Pi, rather than use SCP on the EC2 instance to copy the file over means that the Pi does not need to have a public IP, meaning it is protected from the wider internet, and can remain on a network using NAT.

2.3. Raspberry Pi Driver and Application

The use of the driver and application means that only the set pins will be manipulated, and for a set amount of time when using the toggle function. This prevents any danger of the LEDs or buzzer breaking due to malpractice with the GPIO functions. These programs were reused from a prior assessment and so have been verified to work correctly and as expected, and so were deemed appropriate for this project.

2.1. Cloud

The EC2 instance is using standard free settings, running Ubuntu version 18.04. A security group was created to allow for SSH access for configuration and operational purposes, HTTP access for the web server, and port 8883 for the MQTT broker. The appropriate private key is needed for SSH access to the server, so that only users with the private key can access, resulting in no unauthorised access. The web server is configured to use HTTP rather than HTTPS. Whilst this is less secure, and data is transmitted without encryption, there was no way foreseeable to enable encryption without a registered domain name, and since this project is a proof of concept, it was deemed unnecessary to purchase a domain name. If this project were to be taken further, and deemed commercially viable, then a domain name could be purchased and encryption enabled by acquiring a TLS certificate from an approved certificate authority. The user inputs are not sanitised as the only processing done with them is to save them to a file, and as this file is partially written by a hard coded statement, there is no risk of unnecessary code execution. The MQTT broker disallows unauthorised connections, only accepts encrypted communications, and requires a username and password to connect.

3. Conclusion

3.1. Summary

The project works well and is an ideal tool that can be used by both users wishing to monitor their own network as well as vulnerability testers who test various networks and need a device to scan said networks. There were no main problems encountered, apart from the lack of any clear way to use HTTPS rather than HTTP on the web server. For future work to this project, a cleaner web application would be desired, as at its current stage, it is designed for a proof of concept, and so is not very visually appealing. There could also be some more interaction available from the user, such as perhaps an interrupt button on the device to cancel a scan, or the same type of functionality enabled on the web application. Finally, the ability to view previous scans would be beneficial to vulnerability testers as they could review scans they had already completed.

3.2. Results

Each individual objective has been met by this solution. The raspberry Pi runs NMap from a command created by a user on a web application. This command is passed from a seperate EC2 instance from the Pi, to the Pi. The Pi will display a standby LED, a scanning LED, and sound a buzzer on scan completion. The results from the scan can be viewed on the EC2 instance.

4. References

Lyon, G. (1997). Nmap: the Network Mapper - Free Security Scanner. [online] Nmap.org. Available at: https://nmap.org [Accessed 17 Dec. 2019].

Veillard, D. (1999). The XML C parser and toolkit of Gnome. [online] Xmlsoft.org. Available at: http://xmlsoft.org [Accessed 17 Dec. 2019].