Hogeschool van Amsterdam

Trainee Manual

Hackable ESP Device

IoT Cyber Security Course
5 Dollar ESP Controller
ESPinoza - 24/01/2022
Version 0.1

Hogeschool van Amsterdam

Team:
Thijs Takken
Luke de Munk
Christina Kostine
Twenne Elffers

Table of contents

Version history

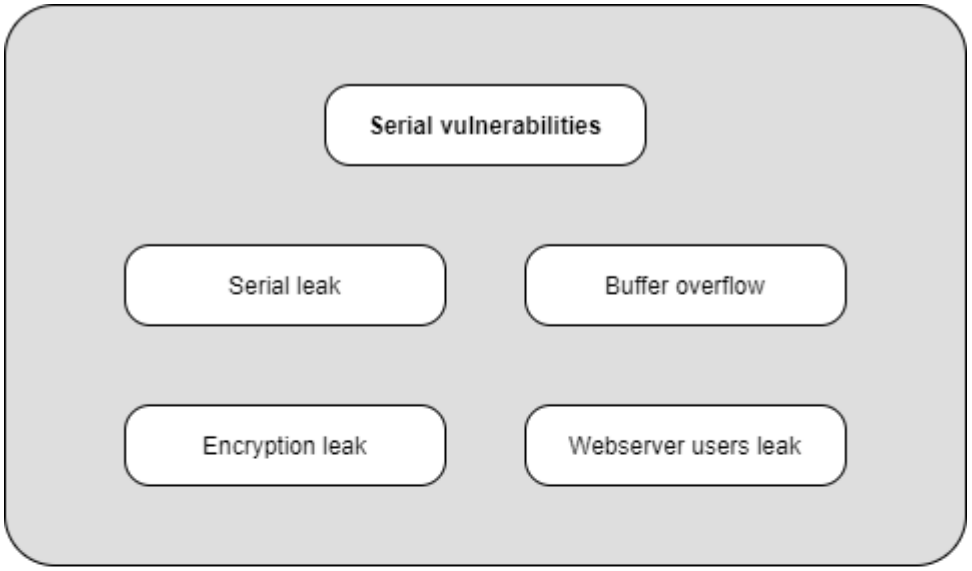
Version	Changes	Date	Author
0.1	Initial version	10-9-2021	Team

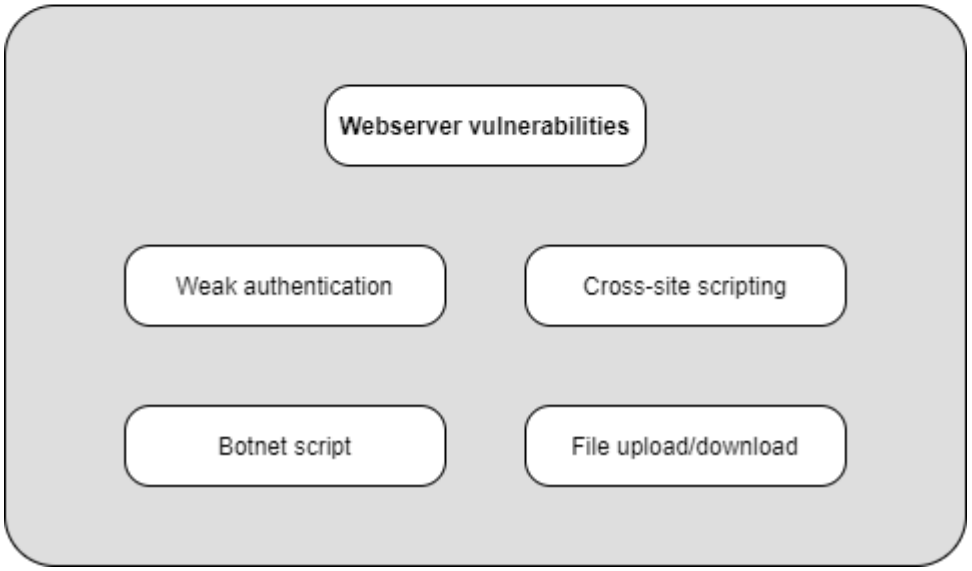
Introduction

Welcome to the Eurofins IoT Cyber Security Training. This document gives insight into the number of vulnerabilities and challenges. You can use this guide to get started with hacking the device, since it contains descriptions and hints for all the vulnerabilities. However the literal answers are not in this document, it is recommended to first try to find vulnerabilities by just exploring the device. By this way, you can see how big your knowledge already is.

When you are stuck or think you have all the vulnerabilities, you can take a look at this guide to make sure you have everything. The first chapter describes all the vulnerabilities on the device abstract and in the second chapter, you can find hints for each vulnerability.

Overview of the vulnerabilities





Description of the vulnerabilities

Serial vulnerabilities

Serial leak (& web server users leak & encryption leak)

The serial console is often used by testing software on microchips. The debug messages can contain sensitive data. This device has the capability to interpret several commands as well. The serial is in this case not turned off before the deployment, which can lead to serious security risks. Especially when commands can be executed, the data on the device is not secure.

Buffer overflow

In the past, buffer overflows were common. Today, most of the software is written in a way that a buffer overflow is not possible anymore. In microchips such as the ESP device, buffer overflows can still exist. A buffer overflow (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. As a result, the program attempting to write the data to the buffer overwrites adjacent memory locations.

For example, a buffer for log-in credentials may be designed to expect username and password inputs of 8 bytes, so if a transaction involves an input of 10 bytes (that is, 2 bytes more than expected), the program may write the excess data past the buffer boundary.

If the transaction overwrites executable code, it can cause the program to behave unpredictably and generate incorrect results, memory access errors, or crashes. If attackers know the memory layout of a program, they can intentionally feed input that the buffer cannot store, and overwrite areas that hold executable code, replacing it with their own code. For example, an attacker can overwrite the program-pointer (an object that points to another area in memory) and point it to an exploit payload, to gain control over the program.

Encryption Leak

CBC mode is an AES block cipher mode that is XOR-ing the first plaintext block with an initialization vector before encrypting it. The decryption works in the same way with ciphered text.

Even if Cbc mode is more secure than EBC one as it hides the patterns of plaintext. This vulnerability is about finding the key and decrypting the file with it.

Web server vulnerabilities

Vulnerable upload

There is a configuration file hidden somewhere on the web server. This configuration can be back-uped an restored. The restoring and back-uping has not been properly secured. Try exploiting it.

Cross-site request forgery

Through the beautiful webpage, users can control their LED controller. But how does this work? Maybe it’s controllable?!

BotLED

Let’s take over all the LED controllers! For this assignment use the provided **BotLED_skel.py** file. Basic knowledge on Python is required.

Hints

IMPORTANT! On the next pages you can find hints. When you have not explored the device yet, it is recommended that you do that first!

Serial vulnerabilities

The list below tells how to get started with the serial vulnerabilities.

1. Connect the device via the USB or the GPIO pins
2. Research common baud rates and try some
3. Research the possible commands

Serial leak

1. Enable the full debug mode by a command
2. Reboot the device
3. Look into the startup bytes

Web server users leak

1. Become a superuser in one of the two ways
2. Enter a command to list all the web server users

Encryption leak

CBC mode is an AES block cipher mode that is XOR-ing the first plaintext block with an initialization vector before encrypting it. The decryption works in the same way with ciphered text.
Even if Cbc mode is more secure than EBC one as it hides the patterns of plaintext. This vulnerability is about finding the key and decrypting the file with it.

1. Become a superuser in one of the two ways
2. Look at the commands

Buffer overflow

1. Look at the possible commands
2. Look into the files with a command
3. The program pointer needs to be overwritten by the overflow
4. Important lines in program file:
    a. 17, 19
5. Important line in disassembled program file
    a. 1, 27
6. Try to run the program with arguments, it takes some tries
7. The way of inputting hexadecimals is, for example: “\x11”

Web server vulnerabilities

Vulnerable upload

There is a configuration file which is used for the webserver. There are three actions that can be executed to exploit two information pieces.

1. Try the “user” page
2. Have you looked into common weak passwords?

1. Have a look at “conf.txt”
2. Look for the admin password
3. Maybe try looking in the serial interface for a way to decrypt the file?
4. gpg –decrypt key string is the correct command
5. The password is located between the double dots

1. There is an admin page with an extra feature
2. Try uploading your own file?

Cross-site request forgery

The main webpage has a function to set the power state. This controls what the LED does.

1. Inspect the set_power function
2. How do you interact with it? Use 0 or 1
3. Create an html document in which you call the right url, this way, whenever a victim uses your webpage, in theory, you as the hacker can control their LED controller

BotLED

The LED controller has a callable function with which you can control the LED. In this challenge it is the goal to control a whole group of ESP’s on a network and manipulate their LED state.

1. Use the URL Library (urllib)
2. Use it to do a request with urlopen
3. Leverage the variables “state” and “addr” in the url to call the devices with
4. Remember, the code already does the loop, you only have to worry about making a single call, no other logic required