Network Security Project

Malik DAHMANI

May 28, 2024

Contents

1 Introduction

CVEs (Common Vulnerabilities and Exposures) are a public listing of computer security vulnerabilities maintained by the MITRE organization. They provide a standardized method for identifying, tracking, and referencing potential vulnerabilities in software and computer systems. A CVE list entry includes a CVE identifier (for example, CVE-2023-38545), a brief description of the vulnerability or security hole, and references, including links to reports and advisories related to the vulnerability. In addition, these vulnerabilities can be rated by severity, from low to critical.

2 CVE-2023-38545

2.1 Introduction

CVE-2023-38545 is a flaw in cURL and is reported to be the worst vulnerability ever discovered in cURL. To begin with, cURL (client URL request library) is a command line interface for transferring data over a network. This data is referred to as a URL.

CVE-2023-38545 is a heap-based buffer overflow vulnerability in cURL, affecting versions 7.69.0 through 8.3.0. A heap-based buffer overflow vulnerability occurs when a program writes more data to a heap-allocated buffer than the buffer is designed to hold. This can lead to a buffer overflow that overwrites adjacent memory and corrupts data.

The vulnerability can be exploited if the cURL client is configured to use a SOCKS5 proxy when connecting to a remote site. To exploit the vulnerability, the attacker must first manipulate the length of the URL so that it exceeds the allowed size (>255 bytes in length) and the connection is slow enough for the bug to occur. If the hostname is detected as longer than 255 bytes, curl switches to local name resolution and passes only the resolved address to the proxy. A slow connection can cause a bug where the local variable responsible for instructing the host to resolve the name may receive an incorrect value. This causes the entire excessively long hostname to be copied to the destination buffer instead of just the resolved address, creating a heap-based buffer overflow. Successful exploitation of this vulnerability could allow an attacker to: execute arbitrary code on the affected system, access sensitive information, take control of the affected system.

2.2 How to reproduce the CVE environment

To reproduce the CVE2023-38545 environment, I need to install a compatible version of cURL, i.e. versions 7.69.0 up to and including 8.3.0, and configure it to use a SOCKS5 proxy.

2.3 How to prepare to reproduce the exploitation

To reproduce this exploit, after configuring cURL correctly, I'm going to set up an http server with a fairly long name. I'll then create a script that requests this server and uses the SOCKS5 proxy, which should trigger the overflow.

3 Reproducing CVE-2023-38545

3.1 The environment

To reproduce the environment required to exploit CVE-2023-38545, we must first ensure that we have a compatible version of curl and libcurl, i.e. versions 7.69.0 up to and including 8.3.0. In this report, I'll use version 7.81.0 of curl and libcurl to illustrate the process. Once this version is installed, you need to create a server using a SOCK5 proxy. This command establishes an SSH connection to the local server (127.0.0.1) and opens a socks5 proxy on port 10801.

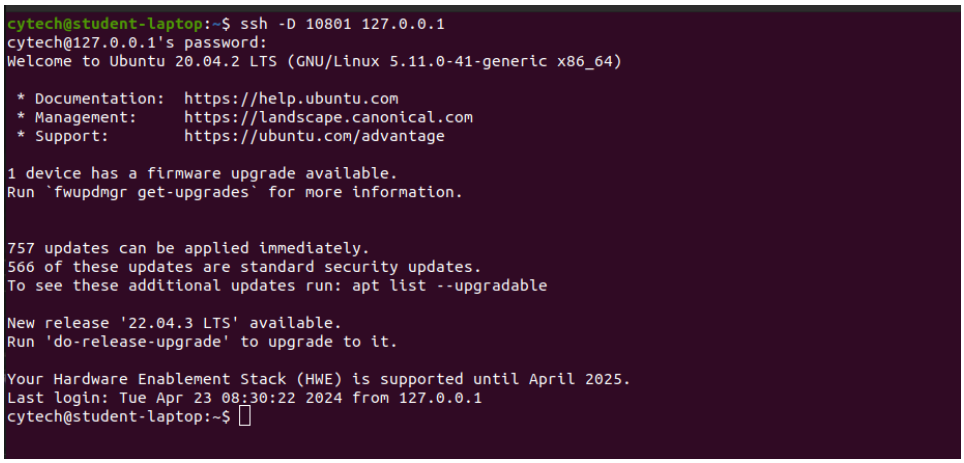

Figure 1: Picture showing the SSH connection to the local server

3.2 The server

Next, I use a Python script to create a server that simulates HTTP responses. This server responds to HTTP requests with a status code of 301, i.e. a redirect response pointing to a very long URL. This server also introduces a 5 second delay to mimic a delayed response. Next, it reads the data to be returned from a text file called "server.txt". Each line of this file is processed separately, and after removing spaces at the beginning and end of the line, the server sends them one at a time, with a 1 second interval between each transmission. This can be used to simulate the progressive sending of data by the server. Finally, two empty lines are sent to mark the end of the server response.

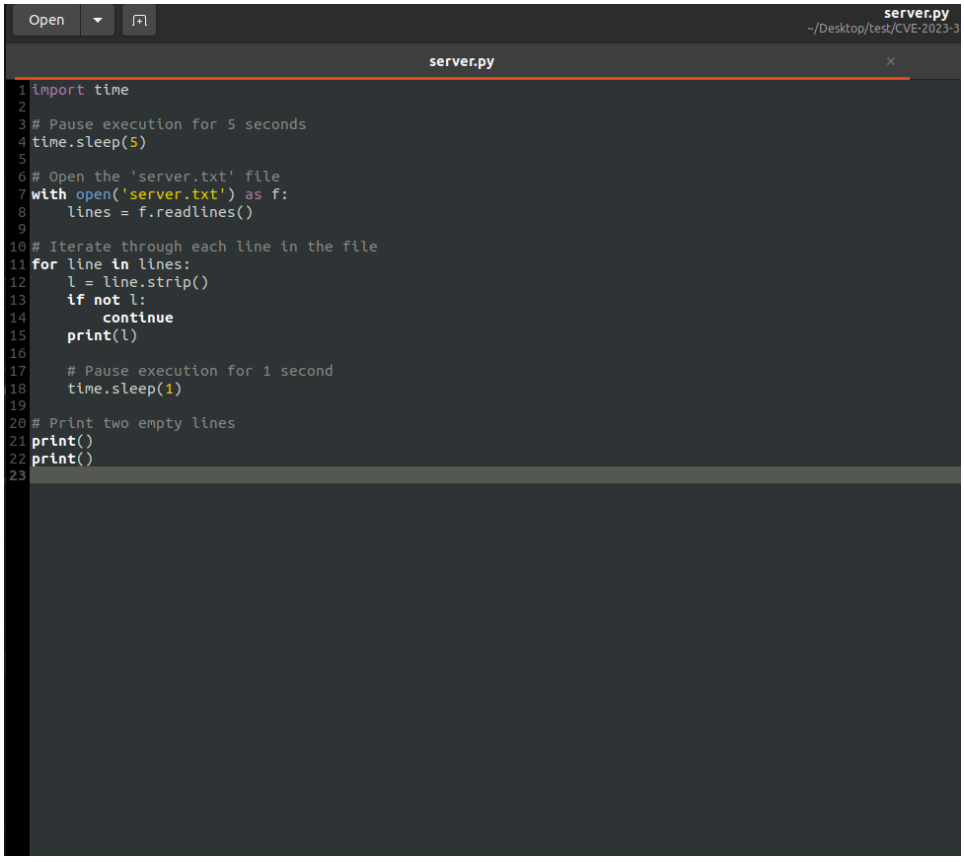

Figure 2: Picture showing the code of the server

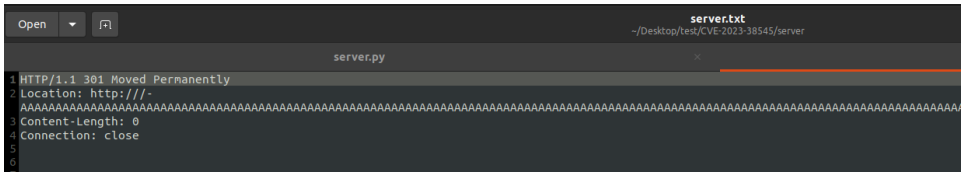

Figure 3: Picture showing the code of the server.txt

To continue, I use the command "python3 server.py | nc -C -nvlp 8000" to run the previous script and redirect its output to netcat(nc) listening on port 8000. The options used are:

- C: Enables data compression during transmission.
- n: Disables DNS resolution, displaying IP addresses in numerical form.
- v: Activates verbose mode, which displays more information about incoming and outgoing connections.
- l: Listen on the specified port (8000 in this case) for incoming connections..
- p 8000: Specifies the port on which netcat should listen.

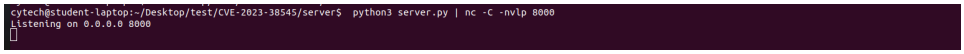

Figure 4: Picture showing the server running

3.3 Exploit CVE-2023-38545

Finally, I use C code that uses the libcurl library to make an HTTP connection through a SOCKS5 proxy to my "http://127.0.0.1:8000/" URL. After configuring the transfer session parameters, such as receive buffer size, destination URL and connection details, the program executes the HTTP request. Once the request is complete, the resources allocated to the transfer session are released to avoid memory leaks. The execution return code is then returned to indicate the success or failure of the operation.

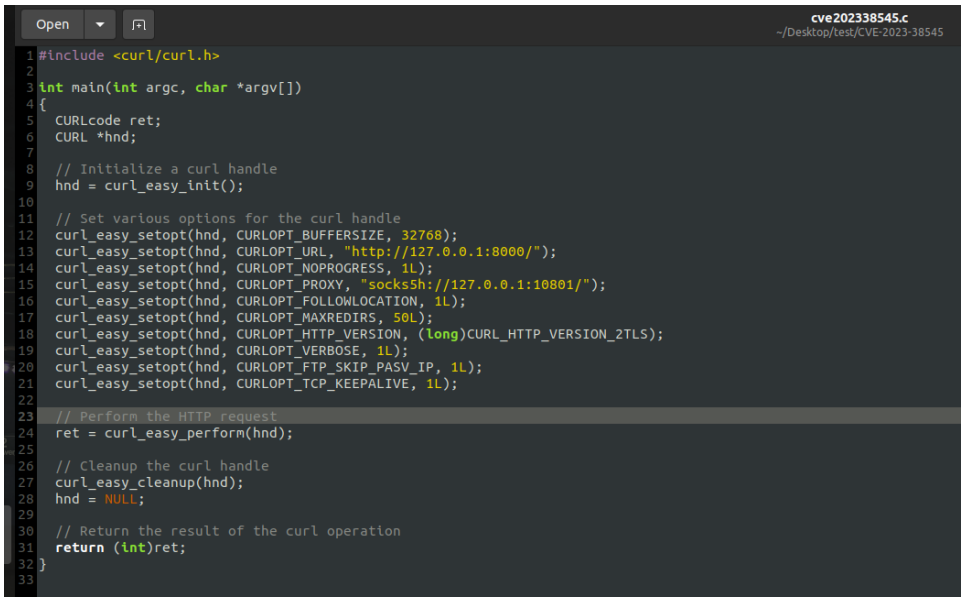

Figure 5: Picture showing the code of cve202338545.c

I used the commands: "gcc cve202338545.c lcurl" then "./a.out" to run this script. The results of the program execution show a segmentation error, suggesting a heap buffer overflow.

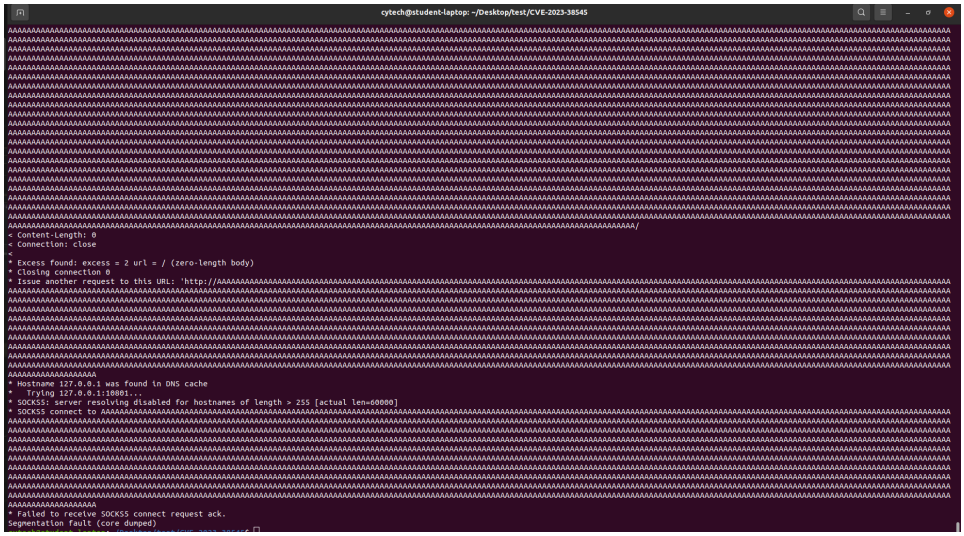

Figure 6: Picture showing the exploitation of CVE-2023-38545

3.4 Github repository

This exploit was inspired by UTsweetyfish github repository available at:`https://github.com/UTsweetyfish/CVE-2023-38545`

4 The log of the attack

4.1 The sock5h proxy server before and during the attack

Before the attack, the server was listening on 0.0.0.0 on port 8000, indicating that it was prepared to accept connections from any available network interface on the specified port.

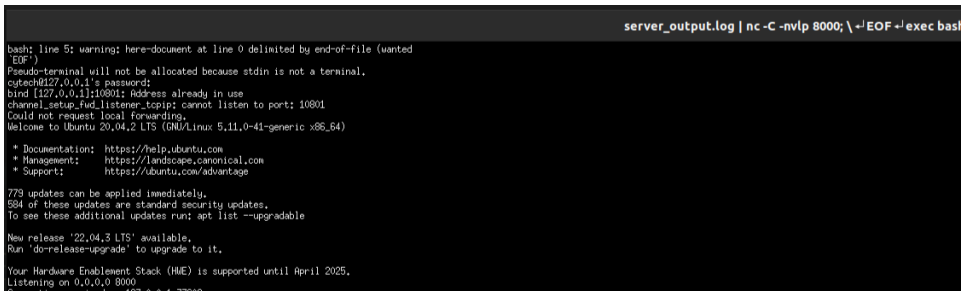

Figure 7: Picture showing the server before the attack

During the attack, the server received a connection from 127.0.0.1 on port 33608, with a GET request for "/". The request was made using HTTP/1.1 protocol, specifying the host as 127.0.0.1:8000 and accepting any type of content.

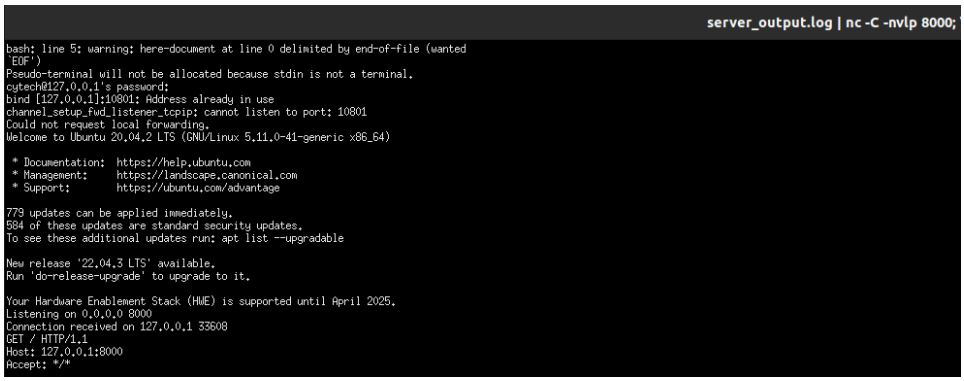

Figure 8: Picture showing the server during the attack

4.2 The result of the attack

Figure 9: Picture showing the result of exposure

The results of the program execution show a segmentation error, suggesting a heap buffer overflow.

4.3 The log of the attack

To retrieve the logs of this attack, I used syslog. Then, in order to specifically target the logs associated with the attack, I used the "cat syslog | grep curl command". This allowed me to filter the entries in the syslog file and isolate those related to the use of curl, making it easier to identify the relevant log.

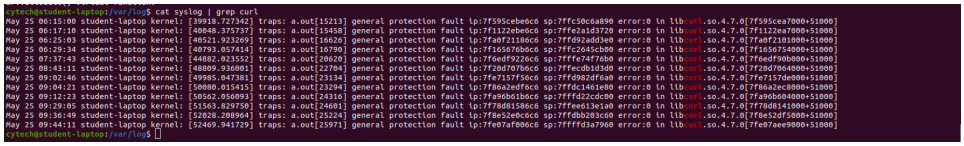

Figure 10: Picture showing the log of the attack

In these logs, the log entry highlighted a "general protection error" at a particular memory address in the "libcurl.so.4.7.0" library. The error, which occurred at memory address "ip:7f595cebe6c6", indicates a memory access violation within the curl library, pointing to a heap buffer overflow.