Metasploit framework exploit module CVE-2021-42013

COMP.SEC.300-2021-2022-1 Secure Programming

Luukas Lusetti – 255162

General description

The aim of the exercise was to study CVE 2021-42013 and create a Metasploit framework exploit module that would automate the testing of the forementioned vulnerability. CVE 2021-42013 is a vulnerability for a deprecated version of Apache that potentially allows for path traversal and/or unauthorized remote code execution (RCE).

Because of the nature of the task not being completely aligned with the template items mentioned in the exercise description, I will alter the structure of this report from the one provided to suit this project better. I will first give a summary of the vulnerability, a summary of Metasploit framework and finally suggestions and ideas for improving the module in the future.

Summary of the vulnerability

CVE 2021-42013 was initially thought to only be a path traversal vulnerability. It was found that an attacker could use double encoded path traversal characters ("../"-characters) to break the path normalization checks performed by httpd. Additionally, it was discovered, that if the server had CGI modules enabled, this vulnerability could be leveraged into an unauthorized RCE. While default httpd configurations prevented this vulnerabilities exploitation both for path traversal and RCE, it was observed to be exploited in the wild. This vulnerability applies only to httpd versions 2.4.50 and 2.4.49, not earlier versions.

Metasploit framework

The Metasploit framework is a Ruby-based penetration testing platform that allows users to create modules used to execute exploit code. It contains a large variety of tools that can be used to test vulnerabilities, execute attacks, scan networks and evade detection.

At its core, Metasploit is composed of modules that are pieces of software that are used to perform specific actions, for example exploiting. Modules are of several different types:

- Exploit, used to take advantage of a vulnerability to provide access to target system. These include web application exploits, code injections and web application exploits.
- Auxiliary, used to perform arbitrary actions that do not execute payloads. These can be denial of service, scanners and fuzzers.

- Post-Exploitation, used to gather information about the target system.
- Payload, used to define how to connect to the shell and what to do once control has been taken.

Another core component of the Metasploit framework is its datastore. The datastore is table of named values that is used to configure the behavior of different components of the Metasploit framework. The "set"-command can be used to define a module level datastore option, which is often used when running Metasploit modules.

Creating and using an exploit module

Creating an exploit module for Metasploit is relatively simple, although it does have a bit of a learning curve if unfamiliar with Ruby.

A large chunk of the module is defined in the metadata section of the module. Here we give different specifications regarding our module such as info related to the vulnerability, target platforms and architectures, default values for options and notes regarding the module/exploit itself. Additionally for this module, I set a default "check module" that can be used to check if the target is vulnerable in the first place before executing the exploit (Metasploit already had a scanner for this CVE). The exploit itself is called by the framework using "def exploit"-function as an entry point.

Once the module has been created, it is rather simple to use. First the module is loaded by calling

"use <path>"

on the Metasploit framework- console where the path is the location of the newly created module. In my case, since the exploit was stored inside ~/.msf4/exploit/test/cveapache.rb, I will set the module to be used by calling

"use exploit/test/cveapache".

Now that the module is loaded, we need to set options that define the target. This can be done using

"set RHOSTS <target_ip>"

"set RPORT <target_port>".

A payload can be chosen from those defined as compatible using

"show payloads"

"set payload <payload_id>"

Now with all options set, we can exploit the vulnerability on the specified target. When running "exploit" a check will be made first to see if the target is vulnerable, after which the exploit code defined in the module is run. Depending on which payload is set, a different type of reverse shell/meterpreter is opened and you should have access to the target system.

Possible improvements for the future

In terms of possible improvements that could be implemented to make the module even better, the module could be implemented so that it also allows for exploitation of path traversal vulnerability mentioned in the first section of this report. Right now, my implementation only allows for exploitation of vulnerable Apache servers with cgi-bin enabled, which is not necessary for path traversal. This could be implemented using options where the user would set the exploit that he wants to use. Additionally, the printouts of the module could be made more clear and concise.