Hacker System Monitor
Matteo Bordignon, 05/03/2024.

Background

The system is vulnerable to Remote Command Execution (RCE) more precisely to Remote OS Command injection[1]. Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation[2].
Possible mitigations:

- use library calls rather than external processes to recreate the desired functionality;
- if possible, ensure that all external commands called from the program are statically created;
- use white list for commands: assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does;
- use allowlist: run time policy enforcement may be used in an allowlist fashion to prevent use of any non-sanctioned commands;
- use proper escaping functions;
- use sandboxes: execution environments in which code can be run in a limited environment;
- more mitigations[3].

Vulnerability

The server responds with a number if the provided input is a legit process name and with a “Timeout” string when illegitimate strings are provided.
Providing a payload such as; $(sleep 5), the server executes the command and responds after 5000ms. So with this proof I thought that the application could be vulnerable to OS command injection.

Solution

I started locating the file flag.txt. To do that I opened a url encoder and wrote this payload:

```
; if [ -f flag.txt]; then sleep 5; fi
```

then I copied the url encoding and pasted into the GET request to the server using the Burp suite repeater tool. Since the server has responded after 5000ms the flag.txt file must be in the current directory.
After so many attempts I noticed that the server responded only with numbers (cat flag.txt approach didn’t worked as I hoped) so I searched on the internet for ways to convert character to ASCII numbers, I found many ways to do that, first I tried this payload:

```
;head -c 10 flag.txt | od -An -t d1 | tr -d ‘ ‘ | tr ‘\n’ ‘ ‘
```

It worked but I obtained a sequence of ascii code of the first 10 chars and I was unable to distinguish them so I thought to do it char by char using the following payload and the burp suite intruder for automate the process of send the requests:

```
;char="$(cut -c 1 flag.txt)"; printf '%d\n' "'$char"
```

Once I obtained all the ascii codes I converted them and I was able to retrieve the flag.
I am pretty sure that this is not the best solution but since the vpn doesn’t work I had little time to try something and this is the first solution that worked. [4][5][6]

References

[1] Remote OS command injection: https://beaglesecurity.com/blog/vulnerability/remote-os-command-injection.html#:~:text=Remote%20OS%20command%20injection%20is,interacts%20with%20the%20operating%20system.

[2] Command injection OWASP definition: https://owasp.org/www-community/attacks/Command_Injection

[3] CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'): https://cwe.mitre.org/data/definitions/78.html

[4] Convert char to ascii code in linux bash: https://www.baeldung.com/linux/shell-ascii-value-character#:~:text=Another%20approach%20to%20get%20the,view%20the%20data%20within%20files.

[5] cut command: https://man7.org/linux/man-pages/man1/cut.1.html

[6]How to convert ASCII character to integer in bash? https://superuser.com/questions/597620/how-to-convert-ascii-character-to-integer-in-bash