Lorenzo Cavada Mat. 220502

Third exercise of OffTech about Buffer Overflow. More information about the exercise can be found at this link. The code used in this exercise can be found here.

The flaw.

The main security flaw was represented by the lack of proper input validation in the application, especially were missing the checks on the bounds of some buffers having a fixed size that were prone to receive a larger input than expected causing a segmentation fault. This carelessness made it possible for a malicious user to perform ad-hoc crafted requests which result in a crash of the server and, so, into a Denial of Service.
In particular, the problem was related to the way that the URI and the header were handled. Basically while parsing these two parameters they were assigned into two arrays with fixed sizes. The lack of a proper input validation made it possible that large URI or large header parameters were directly assigned to these arrays causing a segmentation fault whenever they exceed the maximum length allowed.

The exploit.

Exploiting the flaw is quite easy, its indeed only needed to perform a request with the requested URI/header field large enough to cause an overflow. To have a simulation of how the attack may happen simply start the webserver:

```
sudo /usr/src/fhttpd/webserver localhost 8080
```

After that start one of the 3 script called `payload-xxx.sh` (they will all crash the server, `payload-cl.sh` and `payload-ims.sh` will exploit the vulnerability in the header while, instead, `payload-uri.sh` will exploit the vulnerability in the URI). This script will create a file called `payload` containing the actual request that will cause the overflow and the crash of the server. Than, finally, start `exploit.sh` which will take as parameter the port on which the server is listnening and will perform a request with the just created payload causing the crash of the server (be sure to create the payload file in the same folder were `exploit.sh` is executed).

The fix.

The solution to this flaw is pretty straightforward. To avoid any future exploit I proceeded to add some checking whenever a new user input was assigned to an array with a fixed size. For the header vulnerability, I set a maximum accepted length of the parameter of 512 chars which is way enough considering that the only two possible headers are the `If-Modified-Since` and the `Content-Length` which usually do not have more than 100 characters. For the URI vulnerability instead, I've increased the buffer size of the response variable to 8196 char and I applied a check on the requested path length. In UNIX the maximum length allowed for a path is 4096 char so every request exceeding this value will be rejected. A version of the fixed code with the patch is available with this memo.

Thoughts on the breach

The breach was for sure serious, no trace where left from the attacker and anyone able to reach the webserver was able to cause the crash causing many inconveniences to all the users. Other than that was also possible to inject arbitrary code using this flaw giving to a remote attacker the possibility to obtain a remote shell with root priviledge (a more detailed descrption of this attack can be found in the other memo file). For this reason fixing the flaw is not enough to be sure that all the risk are mitigated, a deep analyisis on the server is needed to ensure the lack of backdoor and each file should be checked to ensure that no malicious tampering have been carryed on.