Lorenzo Cavada Mat. 220502

Third exercise of OffTech about Pathname Attack. A description of the exercise can be found here. All the code presented in this recap can istead be found at this link.

The flaw.

The main security flaw was represented by the lack of proper input validation and sanitization of the user input in the application. The memo application naive treated the user input as trusted accessing to whatever file they requested without checking which file was actually going to be printed on the screen. Another problem was related to the permission used to execute the Perl application. The SUID was indeed enabled leading to the execution of the application always with root privilege, privilege that was not strictly required for the normal behaviour of the service.

Test the flaw.

Is possible to check the severity of the flaw by trying to access some sensible file such as `/etc/shadow` that stores the actual user password. Taken for example an user with the possibility to access to the service with a normal browser simply try to type into the URL `http://localhost:8118/cgi-bin/memo.cgi?memo=/etc/shadow`. The application will print the content of the file on the screen giving to the attacker the possibility to access any information stored on the server even the one normally accessible only with root privilge. An automatic simulation of the exploit can be obtained by running the `exploit.sh` script which basically performs a request to the service saving the result in a file called `shadow.txt`.

The fix.

The solution to this flaw is pretty straightforward. To avoid any future exploit I proceeded to add some sanitization and some checks whenever a new user input was read. To sanitize them I used first the `abs_path()` function and, later, a regex expression to check the validity of the user-provided path.
The `abs_path()` function transform the path passed as a parameter in an absolute path removing all the attempt of escape that can happen using "../". It also checks in the file system if the requested file actually exists. If the first step is correctly carried out the script will start evaluating the absolute path trying to understand if a malicious user is trying to escape from the memo folder. A valid path should point or to `/home/<user>/memo/<file-folder>` or to `/root/memo/<file-folder>`. To do so a regex expression is evaluated blocking the execution of the request in case of a weird path. For applying the patch simply copy the `fixed.patch` file where the `memo.pl` file is stored (`/usr/lib/cgi-bin/`). Rename now the current `memo.pl` to something like `bugged_memo.pl` and, in the same folder, perform the following command.

```
sudo patch bugged_memo.pl -i fixed.patch -o memo.pl
```

This will create a fixed version of `memo.pl` which now will implement a user input sanitization and a check on the path requested by the user. After that is also important to remove the SUID permission from the `memo.cgi` file in order to avoid the execution of it with root privilege avoiding in this way the possibility to access to sensible memory areas such as `/etc/shadow`. To do so perform the following command:

```
sudo chmod u-s /usr/lib/cgi-bin/memo.cgi
```

Thoughts on the breach.

The breach was for sure serious especially because no visible traces was left by the malicious user allowing him to get sensible data such as a password or any other information stored on the server. Is now important to impose a change of the password to all the employee involved in the breach. Is also important that future developer pay more attention to the handling of user-provided input and treat everything as untrusted performing sanitization and proper checking. In this specific case is not enough to just check that a path starts as expected because it can always include special character such as "../" that allows a malicious user to escape from the current folder and move to another with an unexpected result. Other than that is also important to apply the least privilege principle giving to an application just enough privileges to ensure its correct functioning. In this case, the memo service does not need the root privilege that can just lead to unexpected behavior and exploit of the service as the one just fixed.