SSI Project Proposal - G1

SSH1 CRC32 - Compensation Attack Detector Vulnerability

Group members: Adam Gershenson Nogueira, Carlos Rafael Barbosa Madaleno, Diogo Costa Pinto and Ismael Medeiros Moniz

1. Subject

Our mini-project proposal involves developing an interactive approach to learning about integer overflows, by using as a case study the SSH1 CRC32 vulnerability. This vulnerability is related to the SSH1 protocol, and ironically was introduced by a security patch aiming to remedy another vulnerability. The specifics involve a file named *deattack.c,* introduced in a patch to the SSH1 version source code, to detect attacks of the cyclic redundancy check exploit (CRC-32). This patch, while well intentioned, contained an integer overflow in the function *detect_attack*, that could be exploited in a way that allowed for arbitrary code execution in both servers and clients utilizing that patched SSH1 version.

2. Comparing to existing resources

While the SSH1 CRC32 is the target of a fair amount of articles, even going so far as being featured in the Matrix movie, there is no single user-friendly guide on how to exploit it. We intend to provide the user with a hands-on way to learn about SSH1 CRC32.

We also wish to provide baseline knowledge on the way integer overflow vulnerabilities work and their potential risks, as well as show an example of how it can lead to loss of availability of a service. Other labs have touched on these issues, but we aim to explore them in the context of the SSH1 attack and their integration into the lab may help the user explore new ways they may be utilized.

3. Lab plan

Exploiting the SSH1 CRC32 was difficult back in the day and is even more difficult nowadays. There were only two exploits released to the wild, and very little literature about them. Taking this into consideration, we plan on setting up small relevant parts of the original source code of the vulnerable SSH1 version into a small POC that is realistic enough to let the learner understand the core concepts and thought process behind such a relevant exploit.

Firstly, we would start by introducing integer overflows as a general category of exploits, providing some code of our own authorship that could be exploited in a manner that let the learners get some practice.

Then we would familiarize the readers with the specific portion of the SSH1 source code that is vulnerable to an integer overflow, challenging them to crash the program locally and introducing an arbitrary value in a memory buffer.

Finally, in a controlled environment, we would provide them with either a patched SSH client or server that could be sent a payload for realization of the exploit, with the end goal of causing a crash/introducing a value in memory. If it proved too difficult to find compatible SSH clients/servers (this is something prevalent in many of the literature we found, due to how old the exploit is), we would set up a simple stub for a server/client that could receive (unencrypted) payloads and be subject to this exploit, leading to the crashing/introducing of some value in memory of the target program.

4. Tools

Our lab would not require very complex tools. Since the various vulnerable SSH1 implementations were written in C, we need only access to simple programs:

- A suitable C compiler, such as gcc
- gdb, for debugging and showing inner workings of the program memory
- possibly Docker for setting up a controlled and virtual multi-machine environment

5. Conclusion

We believe this specific topic is worth exploring, both due to its historical significance in affecting a protocol deemed secure (and made vulnerable by a security patch of all things!) and due to the lack of clear and easy to follow literature on the subject. We expect it will be a great challenge, but a great case study that should be made accessible to more people, in order to raise awareness about this kind of vulnerability and the overall significance of human error in software security.