Exploits

Simon Le Bail-Collet, Lukas Gelbmann, Björn Gudmundsson

May 2019

Our first three exploits hijack the control flow, while the last two open a calculator.

Exploit 3 (the format string exploit) only works with ASLR disabled. The other exploits with even with ASLR disabled because we compile with the `-static` flag. This flag doesn't make exploiting our code harder (if anything, it becomes easier).

1 Buffer overflow (ping)

The `ping` command uses `sprintf()` on a small buffer when the host is not known. The host name, which is user-controlled, is used as a parameter to `sprintf()`. Since no length checks are performed on the string, the `sprintf()` can overflow the buffer.

We provide a very long host name to overflow the buffer and overwrite the return instruction pointer to hijack the control flow.

2 Buffer overflow ("Command not found")

When formatting the exception description for the case where an incoming command is not specified by the GRASS protocol, no length checks are performed. `sprintf()` is used in a similar manner as in ping and an easy buffer overflow exploit can be performed.

We provide a very long invalid command to again overwrite the return instruction pointer.

3 Format string (ls)

We were able to exploit our format string bug a day after handing in our code. Inexperienced developers as we are, we "accidentally" used `sprintf()` with a user-controlled format string. This allow us to write to an arbitrary address. With ASLR disabled, we can reliably overwrite the return instruction pointer.

4 Command injection (grep)

This exploit takes advantage of how our implementation of the GRASS protocol uses the `ls` system command to traverse all child directories of the current directory. The `grep` command does not surround its argument with single-quotes, allowing for arbitrary command execution.

The exploit goes as follows: After a successful login, the user issues a `mkdir` command and names the directory something with a semicolon, followed by a shell command. Then the user issues a `grep` command in the parent directory, which will run `ls` with a user-provided argument et voilà.

5 Free choice: command injection (rm)

This exploit uses a vulnerability in how directories are made and deleted in our GRASS protocol implementation.

In the `mkdir` command, we made an implementation decision to remove any single-quote characters in the argument and surround the argument with single-quotes in the name of security.

The exploit takes place in `rm`. It takes advantage of the fact that as carefree developers we use the argument to `rm` in a system call, surrounded by single-quotes but not with single-quotes removed from the argument string.

Before we perform this unsafe call, though, we have a check that the directory exists. In this check, the single-quotes are removed.

The exploit thus goes as follows: After a successful login, the user creates a directory that has a semi-colon and then a command to open a calculator. Then the user issues a corresponding `rm` command that has a closing quote character and allows for arbitrary command execution since the check that the directory exists succeeds.