Software Assurance Tips

A product of the Software Assurance Tips Team[2]

Lane Crawford

Monday 27th May, 2024

1 Sorry Root, You're Not the Boss of Me!

Updated Monday 20th May, 2024

In most Unix-like systems like Linux, the root user can do anything it wants. Everyone knows this. If root wants to edit a file, that file will say whatever root wants it to. If root wants to delete everything inside a directory, that directory is toast. Sure, maybe there are some weird hardware things a super-smart hacker might be able to accomplish to hide something in it,[1, Malicious Hardware] but there's no denying that if root can see the files in the directory, then those files don't stand a chance. With one quick command, root can delete a directory and everything inside it, no questions asked. Well, about that...

During one assurance event, I noticed that the process for updating the boot software for the system was to `rm -rf` the boot directory and then untar a `.tgz` file containing the new boot software into the now-empty directory. This reminded me of something I came across while completing a capture-the-flag (CTF) challenge several years earlier: file attributes. File attributes are meta-data that describe (and, importantly, in some cases modify) file and directory behavior. File attributes are filesystem dependent, and you'll find different attributes depending on what version of OS you're using. Windows has different attributes than Unix, and other Unix-like systems have distinct attributes among them. You'll generally find file attributes for encryption, visibility, and compression (among others). But for our purposes, it's enough to know that most flavors of Linux include an attribute for immutability.

Going back to the pentesting event in question, I realized that I could add my own file to the boot directory (or edit an existing file), make it immutable by issuing the bash command `chattr +i that_file`, and prevent that file from being modified or deleted without first removing the immutability attribute (`chattr -i that_file`). This meant that when root called `rm -rf`, anything possessing the immutable attribute would not be deleted. And since both files and directories can be made immutable, this effectively meant that I could make whatever changes I wanted inside the boot directory and have those changes persist.

And what about the untarring? Normally the `tar` command would overwrite files with the same name; however, any files marked immutable would not be overwritten. In that case, `tar` would throw an error for each of those files, stating there was an error trying to overwrite them. But since everyone knows using root to `rm -rf` removes everything, the output from tar was not being checked to ensure it completed successfully. The result was that I could make whatever changes I wanted inside the boot directory, have those changes persist through the update process, and the update process would return a message that the update happened successfully.

Hopefully this will help you realize the importance of understanding little-known Linux attributes and not assuming that simple functions that appear to be straightforward are actually completing the way you want them to. Functions have return values for a reason; check them before assuming everything worked!

References

[1] Jon Hood. “Side-Channel Attacks”. In: SwATips.com (2021). URL: https://www.swatips.com/articles/20210524.html.

[2] Jon Hood, ed. SwATips. https://www.SwATips.com/.