Bash Privileged Mode

and how it can be used to prevent shellshock

Written by Carter Yagemann

Contributions by Amit Ahlawat

Version 1.0

License

Introduction

In the SetUID Lab[1], we saw an example of how unsafe it can be to set the program's user ID equal to the user's effective ID. In the case of that lab, if a C system call is made while the user ID is escalated to the same level as the effective ID, then the program becomes susceptible to the shellshock attack. The question you might have though is if this situation is really plausible in the real world. Would a developer ever want to escalate both the user and effective user IDs?

The Problem

Frankly, yes. The reason has to do with how bash handles privilege. When a developer writes a program which, over the course of its execution lifespan, changes its effective user ID, it is doing so because the program needs to temporarily escalate or deescalate its permissions. It might need to do this in order to perform actions which the caller of the program cannot normally do, like access the system's password file so the user can change their password.

The problem, however, is that when a bash process is started, it checks the user ID against the effective user ID, and if they don't match, bash will deescalate the effect user ID down to the user ID. This means that whatever command bash was tasked with executing will not be executed with the escalated permissions the developer might have wanted. This can be really frustrating for the developer and lead

[1] https://bitbucket.org/carter-yagemann/shellshock/src

him to escalating the user ID to match the escalated effective user ID like we saw in the lab scenario.

Remediation

It seems like the developer is stuck with an all or nothing decision. They are forced to either let bash deescalate their effective user ID, or else they have to escalate the user ID which will cause bash to import function definitions and potentially be exploited by shellshock.

This, however, is not the case. Bash has a privileged mode which can be activated by setting the `-p` flag. In this mode, bash will not deescalate the effective user ID, but it will also skip the shellshock vulnerable `parse_and_execute` method. By doing this, the developer can keep the escalated permissions their program requires without becoming vulnerable to shellshock. This could prove to be an alternative remediation technique if patching bash to a latest version isn't a viable option.