Environment Variable and Set-UID Lab

Clayton B. Hodges

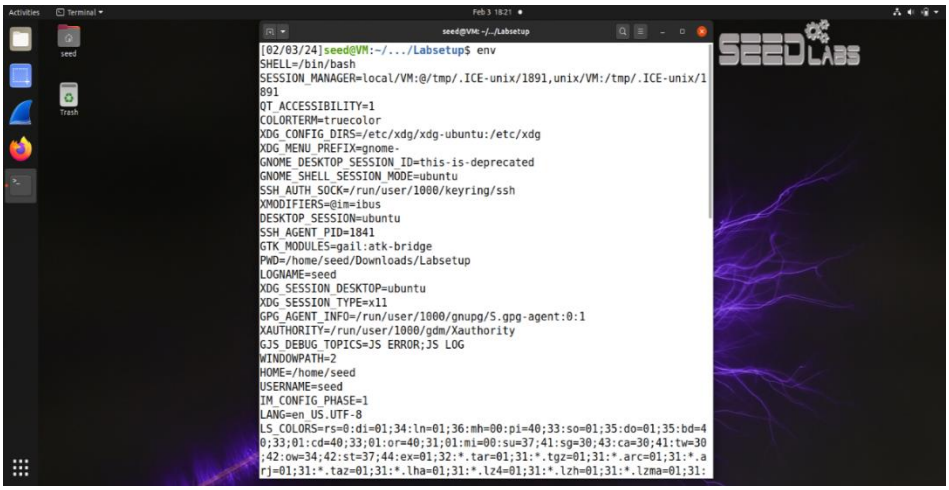
Colorado Mesa University

CSCI 420: Software Security

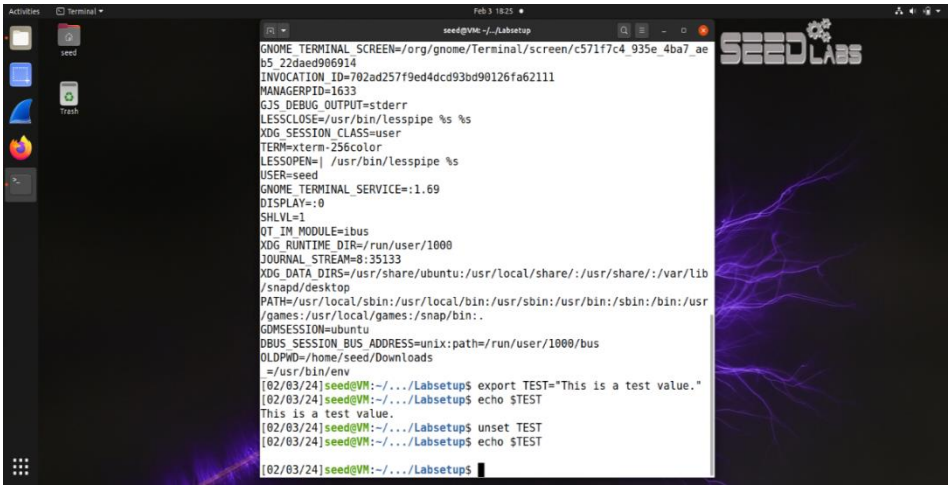
Contents

1. Manipulating environment variables

1.1 Screenshots and Summaries



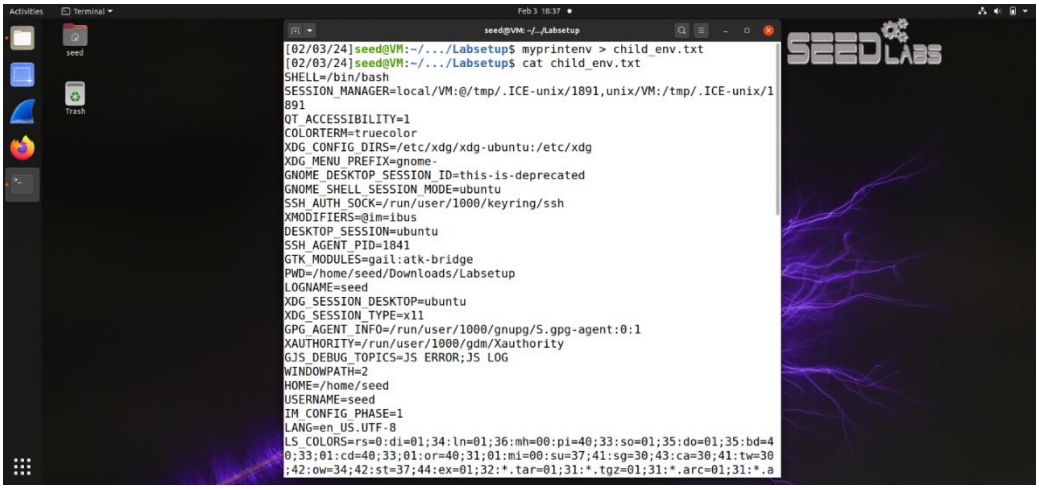
I simply printed out the environment variables using the 'env' command.



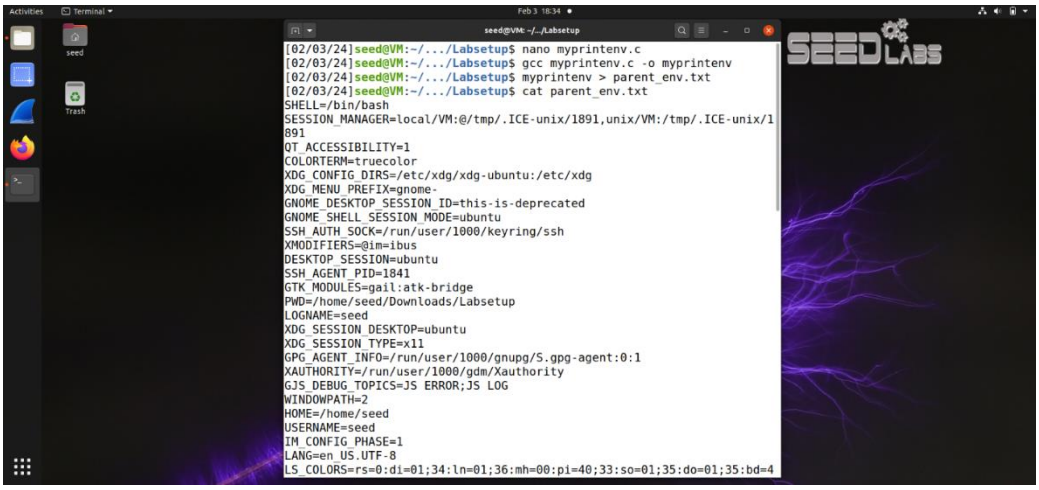
I created and destroyed an environment variable designated 'TEST' using the 'export' and 'unset' commands respectively. I also printed out these changes to the environment variable using the 'echo' command.

2. Inheriting environment variables from parents

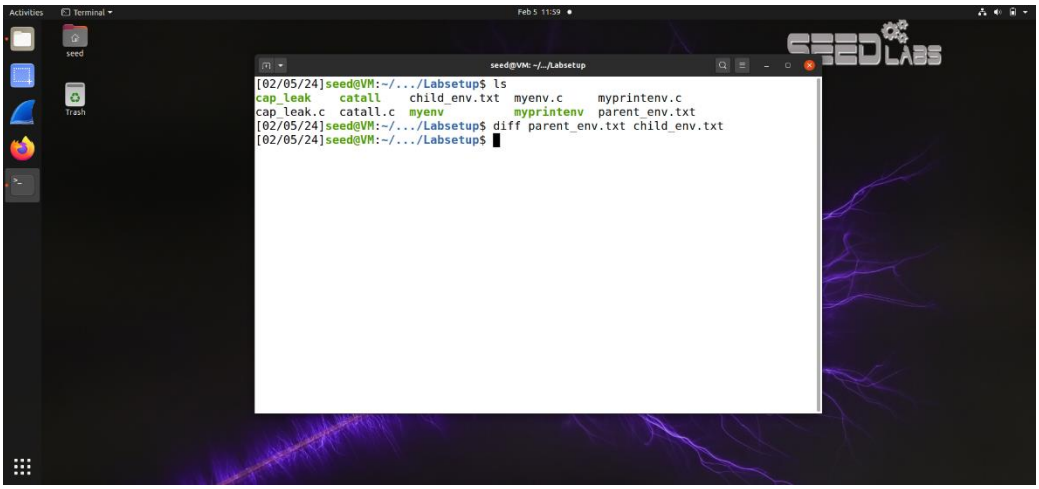
2.1 Screenshots and Summaries



I compiled and ran the provided ‘myprintenv’ program and redirected its output to a text file.



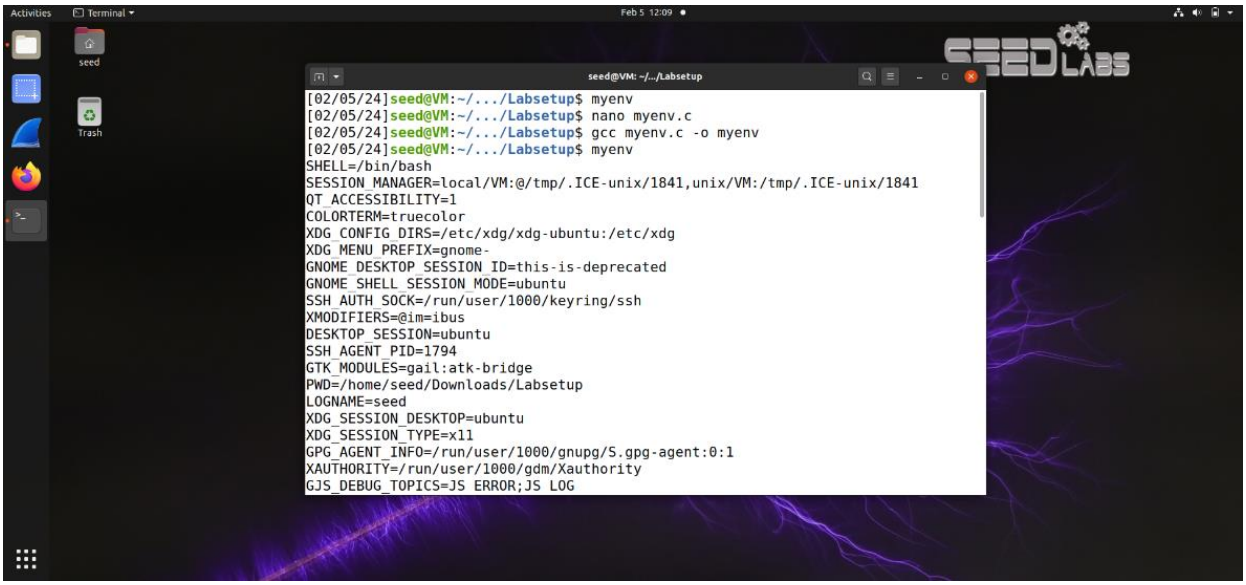
I modified the provided program to print parent process results, noticing no differences in its output text file.



I used the 'diff' command to check for any differences between the two output text files, there were none.

3. Environment variables and execve()

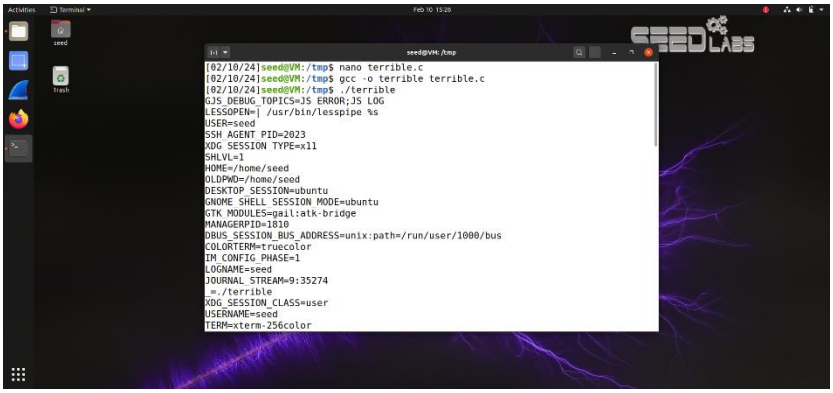
3.1 Screenshot and Summary



I compiled and ran the provided program, initially printing nothing. I then modified the program to include 'environ' in its 'execve' function arguments, printing out the environment variables.

4. Environment variables and system()

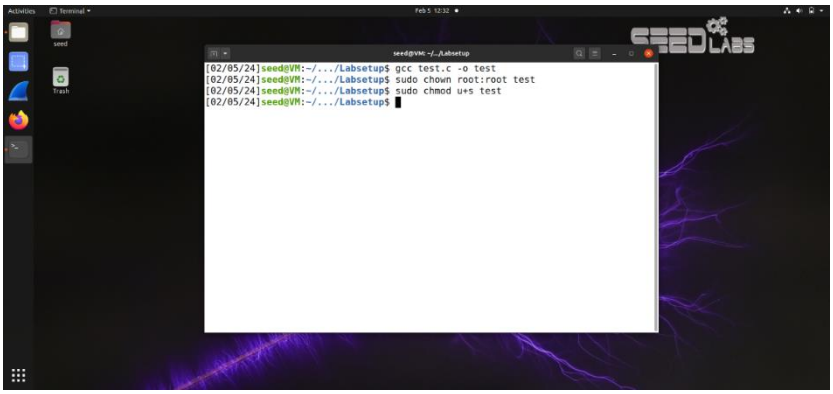
4.1 Screenshot and Summary



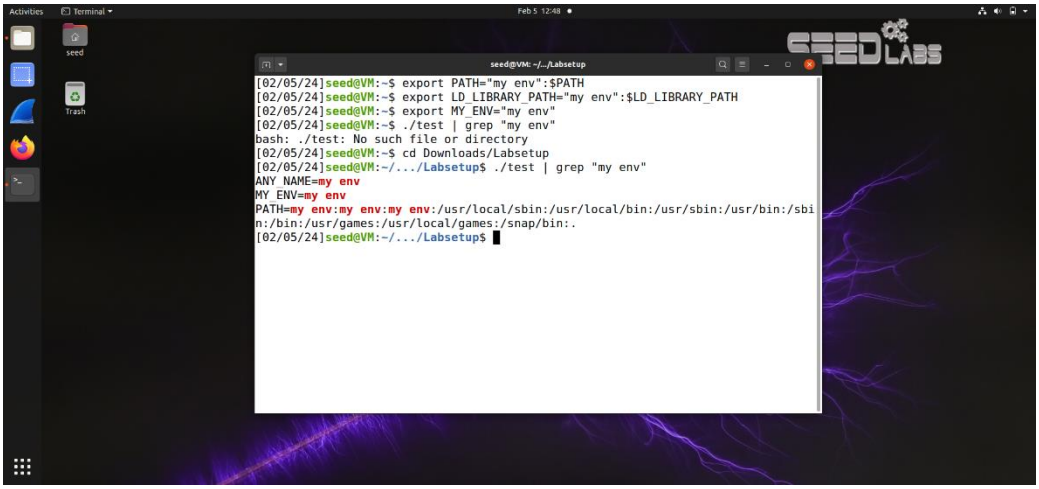
The system("/usr/bin/env") function in the provided program executes the /bin/sh -c command, asking the shell to execute the command. The system() function uses execl() to execute /bin/sh; excel() calls execve(), passing to it the environment variables array and printing them out.

5. Environment variable and Set-UID Programs

5.1 Screenshots and Summaries



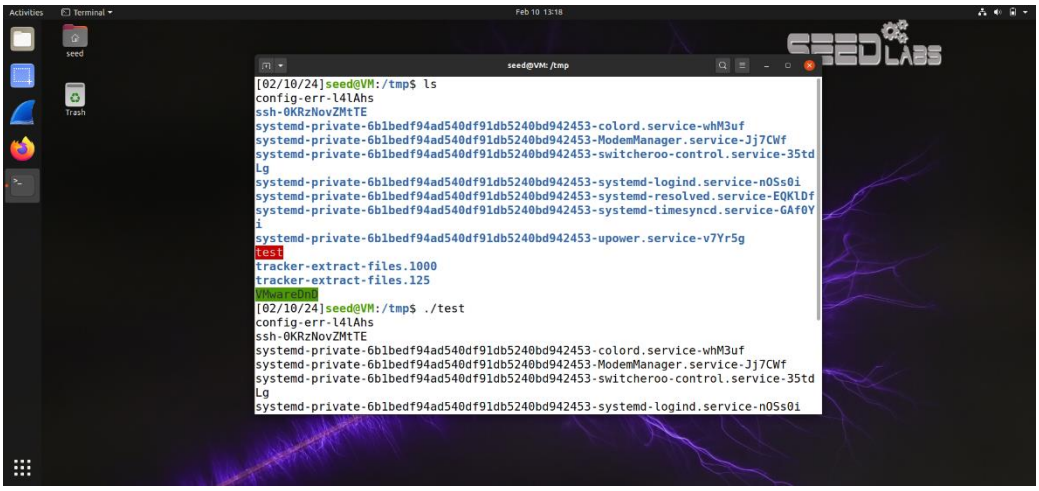
I simply compiled and ran the provided program and set the appropriate root ownership and Set-UID mode.



I proceeded to set the appropriate paths, run the program, and display the corresponding environment variables.

6. The PATH Environment variable and Set-UID Programs

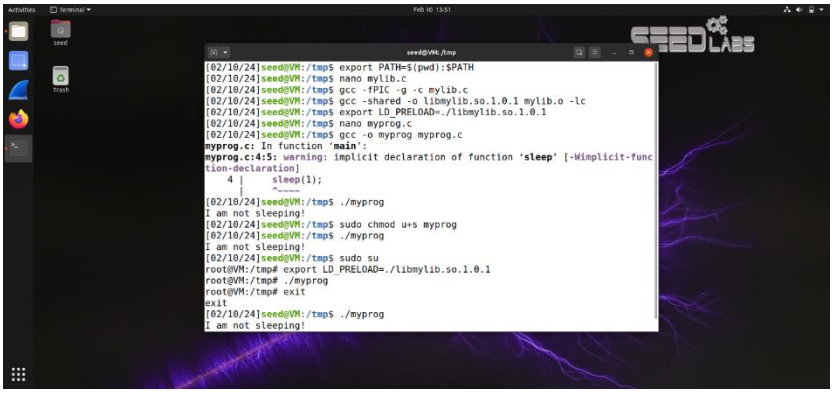
6.1 Screenshot and Summary



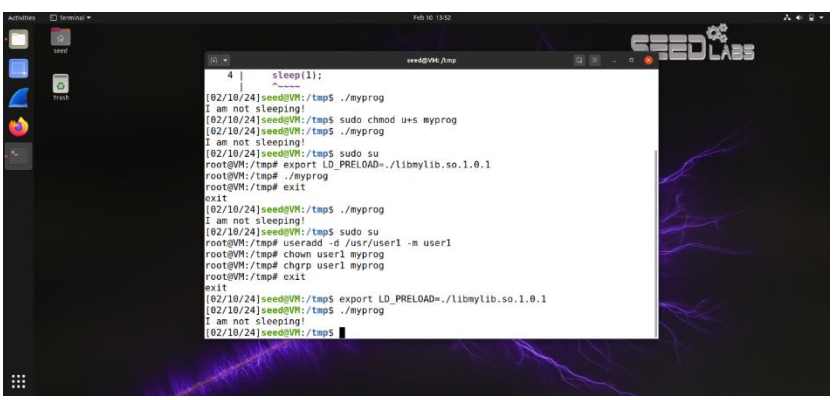
I compiled the provided program, changed the ownership to root, and made it Set-UID. I then displayed the contents of the current directory using the system(ls) function provided in the program.

7. The LD_PRELOAD environment variable and Set-UID Programs

7.1 Screenshots and Summaries



I built a dynamic link library using the provided program mylib.c, compiling and setting the appropriate environment variables. I then compiled and ran another provided function myprog.c that executed the linked program in some cases and myprog.c in other cases. I believe that the ouput makes sense for the regular user and with Set-UID privileges as they are the one who set the environment variable that specifies mylib.c. It also makes sense for no output as root as to prevent unauthorized code from running.



Finally, changing the program's ownership to another user, it still ran which was surprising. I suppose that the child processes inherit LD_* environment variables differently.

8. Invoking external program using system() versus execve()

8.1 Summary

When the program uses system(command) to invoke the command, it's vulnerable to various forms of attacks due to the way system() invokes a shell to execute the command. This shell can interpret shell metacharacters with root privileges due to the Set-UID bit., allowing an attacker like Bob to append additional commands to the input. The same attack wouldn't work if I switched the program to use execve() because execve() does not interpret the input as a shell command but rather as the path and arguments for a single executable.

9. Capability Leaking

9.1 Screenshot and Summary

In this final task, I compiled and ran the provided program as a normal user and was able to write to a root-owned file.