CIS 418/518 – Secure Software Engineering

Environment Variables and Attacks (Chapter 3)

1. What is the difference between environment variables and shell variables?

   Environment variables are global system-wide variables accessible to all processes that run on an operating system.

   Shell variables are transient variables that are internal to the current instance of the shell program running.

2. In Bash, if we run "`export foo=bar`", does it change the environment variable of the current process? How would you verify this?

   "`export foo=bar`" create a shell variable named foo and makes it available as an environment variable to any child processes launched from the shell.

   ```
   $ echo $foo                  // displays the value of shell variable foo

   $ strings /proc/$$/environ | grep foo  // displays the value of foo if the current
                                          // process has foo in its environment variables

   $ printenv foo   ; displays the value of foo if the child process running the printenv command
                    ; has foo in its inherited set of environment variables

   $ env | grep foo     ; displays the value of foo if the child process running the env command
                        ; has foo in its inherited set of environment variables
   ```

3. The following are two different ways to print out the environment variables. Describe their differences.

   ```
   $ /usr/bin/env
   $ /usr/bin/strings /proc/$$/environ
   ```

   `/usr/bin/env` prints the environment variables of the child process.
   `/usr/bin/strings` prints the environment variables of the current process.

4. In our code, when we use `execve()` to execute an external program `xyz`, we pass NULL for the third argument. How many environment variables will the process running `xyz` has?

   Since we are passing NULL as the third argument to `execve()`, the process running `xyz` program will have zero environment variables when it starts.

5. Bob says that he never uses any environment variables in his code, so he does not need to worry about any security problem caused by environment variables. Is he correct?

No. The behavior of a program can be influenced by its use of hidden (not explicitly mentioned or accessed in the code) environment variables.

6. A program `abc` invokes an external program `xyz` using `system()` function, which is affected by the PATH environment variable. When we invoke `abc` from a shell prompt, how does the shell variable PATH in the current shell end up affecting the behavior of the `system()` function?

   When the `abc` program is started from a shell prompt, the shell variable PATH will be passed in an environment variable to the `abc` program which passes it as an environment variable to the child shell process launched to execute the `xyz` program.

7. There are two typical approaches for letting normal users do privileged tasks. One approach is to write a root-owned Set-UID program and let the user run that program. Another approach is to use a dedicated root daemon to do those privileged tasks for the users. Compare the attack surface of these two approaches and describe which one is more secure.

   See Section 3.7 in the textbook for explanation.