

CYSE 211 SEED Labs – Environment Variable and Set-UID Program

Lab

This lab demonstrates how environment variables affect program and system behaviors. In addition, this lab explains how environment variables can affect Set-UID/privileged programs. Additionally, it explains how a child process can be propagated from a parent process.

Task 1

Task: In this task, we study the commands that can be used to set and unset environment variables. We are using Bash in the seed account. The default shell that a user uses is set in the `/etc/passwd` file. `/etc/passwd` contains the default shell that a user uses.

```
environ_set_uid
[02/10/20]seed@VM:~/Desktop$ cd environ_set_uid/
[02/10/20]seed@VM:~/.../environ_set_uid$ printenv pwd
[02/10/20]seed@VM:~/.../environ_set_uid$ printenv PWD
/home/seed/Desktop/environ_set_uid
[02/10/20]seed@VM:~/.../environ_set_uid$ env | grep PWD
PWD=/home/seed/Desktop/environ_set_uid
OLDPWD=/home/seed/Desktop
[02/10/20]seed@VM:~/.../environ_set_uid$ export foo='test string'
[02/10/20]seed@VM:~/.../environ_set_uid$
[02/10/20]seed@VM:~/.../environ_set_uid$ printenv foo
test string
[02/10/20]seed@VM:~/.../environ_set_uid$ unset foo
[02/10/20]seed@VM:~/.../environ_set_uid$ printenv foo
[02/10/20]seed@VM:~/.../environ_set_uid$
```

Observation: This task allows us to understand what each command does. The `printenv PWD` command prints out the environment variables and it shows the exact pathway to our directory. `env | grep PWD` allows us to look at a specific environment (pwd). `export foo` allows us to add a new environmental variable. If we do the `printenv` this will execute the string that we just exported. Unsetting this string removes it from the environment and the execution will go back to its initial result.

Task 2

Task: Task 2 demonstrates how the environmental variables of a parent process are given to a child process. `Fork()`, essentially creates a duplicate process of itself. The child process is the duplicate of the original process, or the parent process.

```

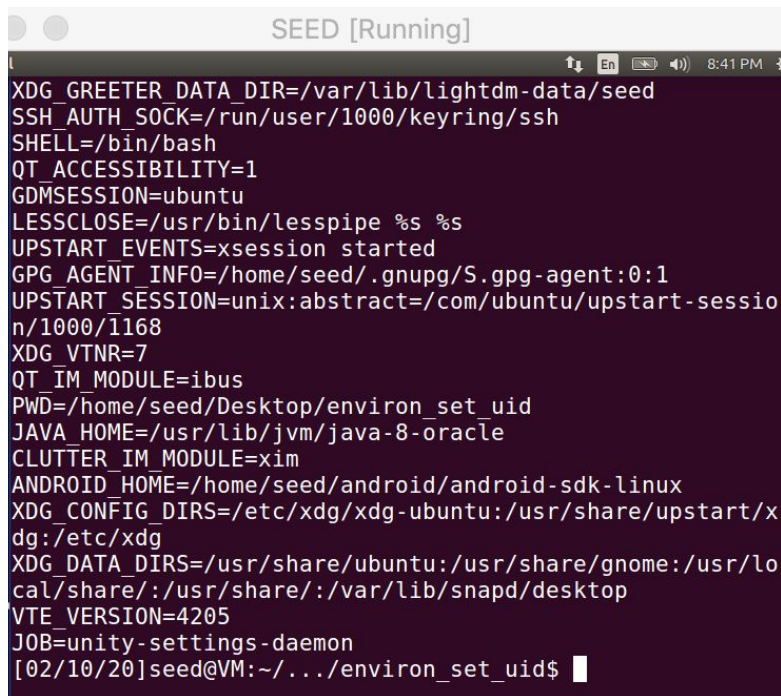
[02/12/20]seed@VM:~/.../environ_set_uid$ diff -y parent
child.
XDG_VTNR=7
DG_VTNR=7
XDG_SESSION_ID=c1
DG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
DG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
LUTTER_IM_MODULE=xim
SESSION=ubuntu
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
NDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
PG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
ERM=xterm-256color
VTE_VERSION=4205
TE_VERSION=4205
SHELL=/bin/bash
HELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db

```

Observation: The first output of the of task2.c is saved into child and the second output is saved into parent. When I first executed diff child parent, nothing came up. So I did 'man diff' to see what error I was making. '-y' provides a side by side comparison of child and parent. It is interesting because it is essentially all the same. However, the first letter of the second one for all of the information is removed. When doing ls -l, they both had the same privileges. The new program gets its variables based on what the user does to the variables within the file.

Task 3

Task: Task 3 demonstrates how environmental variables are impacted when a program is completed by `execve()`.



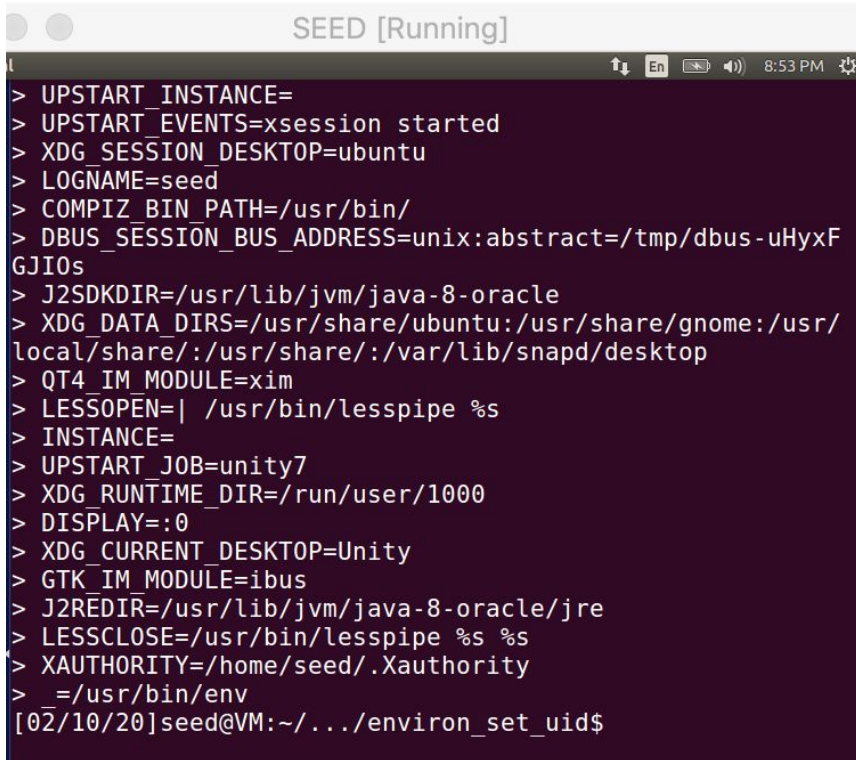
A terminal window titled "SEED [Running]" displays a list of environment variables. The variables are listed in white text on a dark purple background. The list includes XDG_GREETER_DATA_DIR, SSH_AUTH_SOCK, SHELL, QT_ACCESSIBILITY, GDMSESSION, LESSCLOSE, UPSTART_EVENTS, GPG_AGENT_INFO, UPSTART_SESSION, XDG_VTNR, QT_IM_MODULE, PWD, JAVA_HOME, CLUTTER_IM_MODULE, ANDROID_HOME, XDG_CONFIG_DIRS, XDG_DATA_DIRS, VTE_VERSION, and JOB. The prompt at the bottom is [02/10/20]seed@VM:~/.../environ_set_uid\$.

```
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1168
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Desktop/environ_set_uid
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share:/var/lib/snapd/desktop
VTE_VERSION=4205
JOB=unity-settings-daemon
[02/10/20]seed@VM:~/.../environ_set_uid$
```

Observation: `Execve()` caused the current program being run by the calling process to be replaced with the new program, and it never returns. This new program has newly initialized stack, heap, and (initialized and uninitialized) data segments.

Task 4

Task: In Task 4, we look at the effect executing a new program with `system()`, has on environmental variables. `system()` is used to execute `"/bin/sh -c command"` asks the shell to execute the command, unlike `execve()`, which directly executes a command.

A screenshot of a terminal window titled "SEED [Running]". The terminal shows a list of environment variables being printed, one by one, preceded by a greater-than sign (>). The variables include UPSTART_INSTANCE, UPSTART_EVENTS, XDG_SESSION_DESKTOP, LOGNAME, COMPIZ_BIN_PATH, DBUS_SESSION_BUS_ADDRESS, J2SDKDIR, XDG_DATA_DIRS, QT4_IM_MODULE, LESSOPEN, INSTANCE, UPSTART_JOB, XDG_RUNTIME_DIR, DISPLAY, XDG_CURRENT_DESKTOP, GTK_IM_MODULE, J2REDIR, LESSCLOSE, XAUTHORITY, and _=. The prompt at the bottom is [02/10/20]seed@VM:~/.../environ_set_uid\$.

```
> UPSTART_INSTANCE=
> UPSTART_EVENTS=xsession started
> XDG_SESSION_DESKTOP=ubuntu
> LOGNAME=seed
> COMPIZ_BIN_PATH=/usr/bin/
> DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-uHyxF
GJI0s
> J2SDKDIR=/usr/lib/jvm/java-8-oracle
> XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/
local/share:/usr/share:/var/lib/snapd/desktop
> QT4_IM_MODULE=xim
> LESSOPEN=| /usr/bin/lesspipe %s
> INSTANCE=
> UPSTART_JOB=unity7
> XDG_RUNTIME_DIR=/run/user/1000
> DISPLAY=:0
> XDG_CURRENT_DESKTOP=Unity
> GTK_IM_MODULE=ibus
> J2REDIR=/usr/lib/jvm/java-8-oracle/jre
> LESSCLOSE=/usr/bin/lesspipe %s %s
> XAUTHORITY=/home/seed/.Xauthority
> _=/usr/bin/env
[02/10/20]seed@VM:~/.../environ_set_uid$
```

Observation: `system()` uses `execl()` to execute `/bin/sh`; `execl()` calls `execve()`, passing to it the array of environment variables. The environment variables of the calling process are propagated to `/bin/sh`.

Task 5

Task: Task 5 shows that a Set-UID program assumes the owner's privileges after it runs. It extends the user's privilege when executed, making it quite unsecure. Environmental variables set by the user can affect the behaviors of Set-UID, along with the program logic.

```
[02/12/20]seed@VM:~/.../environ_set_uid$ sudo chown root task5
[02/12/20]seed@VM:~/.../environ_set_uid$ sudo chmod 4755 task5
[02/12/20]seed@VM:~/.../environ_set_uid$ printenv PATH
/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[02/12/20]seed@VM:~/.../environ_set_uid$
```

```
local/bin
[02/12/20]seed@VM:~/.../environ_set_uid$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
[02/12/20]seed@VM:~/.../environ_set_uid$
```

```
declare -x WINDOWID="65023349"
declare -x XAUTHORITY="/home/seed/.Xauthority"
declare -x XDG_CONFIG_DIRS="/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg"
declare -x XDG_CURRENT_DESKTOP="Unity"
declare -x XDG_DATA_DIRS="/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share:/var/lib/snapd/desktop"
declare -x XDG_GREETER_DATA_DIR="/var/lib/lightdm-data/seed"
declare -x XDG_RUNTIME_DIR="/run/user/1000"
declare -x XDG_SEAT="seat0"
declare -x XDG_SEAT_PATH="/org/freedesktop/DisplayManager/Seat0"
declare -x XDG_SESSION_DESKTOP="ubuntu"
declare -x XDG_SESSION_ID="c1"
declare -x XDG_SESSION_PATH="/org/freedesktop/DisplayManager/Session0"
declare -x XDG_SESSION_TYPE="x11"
declare -x XDG_VTNR="7"
declare -x XMODIFIERS="@im=ibus"
declare -x task5="task5 env variable"
[02/12/20]seed@VM:~/.../environ_set_uid$
```

```
~/usr/bin/printenv
[02/12/20]seed@VM:~/.../environ_set_uid$ printenv task5
task5 env variable
[02/12/20]seed@VM:~/.../environ_set_uid$ export task5='task5 env variable'
[02/12/20]seed@VM:~/.../environ_set_uid$ printenv task5
task5 env variable
[02/12/20]seed@VM:~/.../environ_set_uid$
```



```

UPSTART_INSTANCE=
UPSTART_EVENTS=xsession started
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-uHyxF
JI0s
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/
ocal/share:/usr/share:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
_=/usr/bin/env
02/12/20]seed@VM:~/.../environ_set_uid$ █

```

```

> _=/usr/bin/env
[02/12/20]seed@VM:~/.../environ_set_uid$ man diff
[02/12/20]seed@VM:~/.../environ_set_uid$ diff -q setuidenv_result env_result
Files setuidenv_result and env_result differ
[02/12/20]seed@VM:~/.../environ_set_uid$ █

```

Observation: LD_LIBRARY_PATH is the predefined environmental variable, which sets the path which the linker should look into while linking shared libraries. LD_LIBRARY_PATH contains a colon separated list of paths and the linker gives priority to these paths over the standard library paths /lib and /usr/lib. The standard paths will still be searched, but only after the list of paths in LD_LIBRARY_PATH has been exhausted.

Task 6

Task: Task 6 shows us that calling system() within a Set-UID program is risky, because the behavior of the shell program can be affected by environment variables provided by the user, such as PATH. Users who have negative intentions can manipulate the behavior of Set-UID by changing the variables. The system(cmd) function executes the /bin/sh program first, and then

asks this shell program to run the cmd command.

```
declare -x XDG_RUNTIME_DIR="/run/user/1000"
declare -x XDG_SEAT="seat0"
declare -x XDG_SEAT_PATH="/org/freedesktop/DisplayManager/Seat0"
declare -x XDG_SESSION_DESKTOP="ubuntu"
declare -x XDG_SESSION_ID="c1"
declare -x XDG_SESSION_PATH="/org/freedesktop/DisplayManager/Session0"
declare -x XDG_SESSION_TYPE="x11"
declare -x XDG_VTNR="7"
declare -x XMODIFIERS="@im=ibus"
declare -x task5="task5 env variable"
[02/12/20]seed@VM:~/.../environ_set_uid$ echo $PATH
/home/seed:/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[02/12/20]seed@VM:~/.../environ_set_uid$ ./task6
[02/12/20]seed@VM:~/.../environ_set_uid$
```

Observation: The screenshot above depicts the result of executing 'echo \$PATH'. Echo outputs the strings it is being passed as arguments. It is typically used in shell scripts and batch files to output status text to the screen or a computer file, or as a source part of a pipeline. My code did not run with the Set-UID program.

Task 7

Task: Tasks 7 shows how Set-UID programs deal with some of the environment variables.

There are several environmental variables that affect Set-UID behaviors. LD_LIBRARY_PATH and LD_PRELOAD are the two that we are used in this lab.

```
[02/12/20]seed@VM:~/.../environ_set_uid$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:8:1: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  sleep(1);
  ^
[02/12/20]seed@VM:~/.../environ_set_uid$
```

```
myprog.c:8:1: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  sleep(1);
  ^
[02/12/20]seed@VM:~/.../environ_set_uid$ chmod 4755 myprog
[02/12/20]seed@VM:~/.../environ_set_uid$ ls -l myprog
-rwsr-xr-x 1 seed seed 7348 Feb 12 14:39 myprog
[02/12/20]seed@VM:~/.../environ_set_uid$ ./myprog
I am not sleeping!
[02/12/20]seed@VM:~/.../environ_set_uid$
```

Observation: After changing the privileges of the file, I was able to receive the message 'I am not sleeping'. LD_PRELOAD specifies a list of additional, user-specified, shared libraries to be loaded before all others.

Task 8

Task: Task 8 teaches us that `system()` and `execve()` can both be used to run new programs, `system()` is risky in privileged programs, such as Set-UID programs. The PATH environment variables affect `system()`, because the variable affects how the shell works. `execve()` does not have the problem, because it does not invoke shell.

```
[02/12/20]seed@VM:~/.../environ_set_uid$ vi task8.c
[02/12/20]seed@VM:~/.../environ_set_uid$ gcc task8.c -o
task8
task8.c: In function 'main':
task8.c:19:1: warning: implicit declaration of function
'execve' [-Wimplicit-function-declaration]
^
execve(v[0], v, NULL);
[02/12/20]seed@VM:~/.../environ_set_uid$ █

[02/12/20]seed@VM:~/.../environ_set_uid$ ls -l rootfile
ls: cannot access 'rootfile': No such file or directory
[02/12/20]seed@VM:~/.../environ_set_uid$ █
```

Observation: root file is not a file that exists and therefore the privileges for the file cannot be seen. You cannot remove a file that is not writable to you. However, in the case of Bob (from the scenario), he can remove it because the file is executable for him.