SHELL SCRIPTING

Prabhjeet Singh

Shells

Use different types of shells, by default bash shell will be available on linux systems, but if we want to change that. Use below commands. But these commands will only work if these shells are installed already.

(kali㉿kali)-[~/linux_learning]

└─$ **ksh**

$ ls

(kali㉿kali)-[~/linux_learning]

└─$ **csh**

% ls

(kali㉿kali)-[~/linux_learning]

└─$ **tcsh**

Kali:~/ linux_learning -> ls

–(kali㉿kali)-[~/linux_learning]

└─$ **zsh**

┌──(kali㉿kali)-[~/linux_learning]

└─$

To display environment variables in current shell

$ printenv

SHELL=/usr/bin/zsh

SESSION_MANAGER=local/kali:@/tmp/.ICE-unix/1029,unix/kali:/tmp/.ICE-unix/1029

WINDOWID=0

QT_ACCESSIBILITY=1

COLORTERM=truecolor

XDG_CONFIG_DIRS=/etc/xdg

Variables.

Path variable stores the list of the directories that are searched to find the commands to be executed.

When we first type a command in the shell, it will search if it is a built-in command and if not then it will search in these directories for this command.

If command not found in any where then it won't execute.

IF we put current working directory in this path this is done to make easier to execute the commands and scripts in current directory. However, attacker can use this to execute the trojan.

For example – attacker can add custom ls command in current directory that will be executed instead of the usual ls command. And if we run that with root privileges, we may give full access to attacker.

2 types of Env variable.

1. Global
2. Local

1. Global env variables can be access by any thing executing in that shell.
2. Local variable – if we create a local variable. The script won't have access since it executes in a subshell.

   Technically, global variables called environment variable and local variables called shell variables.

How to create variables.

1. **Local**

   a. Create local variable.
   └─$ COUNT_LOCAL=55

   Print local variable.
   └─$ echo $COUNT_LOCAL
   55

2. **Global, using export command.**

   a. Create global variable
   └─$ export COUNT_GLOBAL=23

   b. Print global variable
   └─$ echo $COUNT_GLOBAL
   23

   Unset or delete the global and local variable, use unset command.

   $ unset COUNT_GLOBAL

   Try to access it.
   └─$ echo $COUNT_GLOBAL
   ➔ Nothing will print as it is unset.

Startup files for Shells.

These file names will be different for different shells.

We can edit “alias” in these files. This is called lazy typing also.

For example, we can make alias of ‘clear’ command as

In startup file type –

alias c = ‘clear’

save the file and type

$ source <startup file name>

It will refresh the setup, ad when we type only

$ c

It will clear the content on the terminal.

While doing ethical hacking or pen testing, so we can add some of alias names which have very long commands so that we can minimize the time to write them on the terminal.

Redirecting input and output with <,2 and & symbols

We have 3 types of files – std in, std out and std errors

```
└─$ ls
combine1.txt   file1_hard.txt   file2.txt  'file   space.txt'   my_dir1   space.txt
file           file1_soft.txt   file3.txt  'file space.txt'     my_dir2   zip__backup.zip
└─$ ls > dir_list.txt
└─$ ls
combine1.txt   file           file1_soft.txt   file3.txt        'file space.txt'   my_dir2     zip__backup.zip
dir_list.txt   file1_hard.txt   file2.txt      'file   space.txt'   my_dir1         space.txt
└─$ cat dir_list.txt
combine1.txt
dir_list.txt
file
file1_hard.txt
file1_soft.txt
```

file2.txt

file3.txt

file space.txt

file space.txt

my_dir1

my_dir2

space.txt

zip__backup.zip

'>' → this symbol will add output of the command to a file.

$ ls -a > output.txt

Here output of the command will be added to the output.txt file.

'>>' → this symbol will append the output of the command to the existing file or if not existing then it will create new file.

$ ls -a /etc >> output.txt

By this we can append data to the output.txt file.

What is difference between $head etc/passwd and $ head < etc/passwd?

Answer -:

$head etc/passwd → In this head utility open the passwd file and displays the first 10 lines.

$ head < etc/passwd → In this, shell took the entire content of the passwd file and send them to the head utility, where head utility displays first 10 lines of the file.

$ find /-name 'sample.txt' 2> errors.txt

{here it will show the sample.txt file path where it found.}

And in errors.txt file it will capture the if there is any error like – permission denied.

$ find / -name 'sample.txt'

find: '/run/udisks2': Permission denied

find: '/run/docker': Permission denied

find: '/run/containerd': Permission denied

find: '/run/lightdm': Permission denied

find: ‘/run/user/1000/systemd/inaccessible/dir’: Permission denied

find: ‘/run/sudo’: Permission denied

find: ‘/run/openvpn-server’: Permission denied

find: ‘/run/openvpn-client’: Permission denied

find: ‘/run/cryptsetup’: Permission denied

find: ‘/run/systemd/unit-root’: Permission denied

find: ‘/run/systemd/inaccessible/dir’: Permission denied

find: ‘/run/initramfs’: Permission denied

find: ‘/usr/lib/mysql/plugin/auth_pam_tool_dir’: Permission denied

/usr/share/inetsim/data/tftp/tftproot/sample.txt

/usr/share/inetsim/data/http/fakefiles/sample.txt

/usr/share/inetsim/data/ftp/ftproot/sample.txt

find: ‘/root’: Permission denied

$ find / -name 'sample.txt' 2> errors.txt

/usr/share/inetsim/data/tftp/tftproot/sample.txt

/usr/share/inetsim/data/http/fakefiles/sample.txt

/usr/share/inetsim/data/ftp/ftproot/sample.txt

/var/lib/inetsim/tftp/tftproot/sample.txt

/var/lib/inetsim/http/fakefiles/sample.txt

/var/lib/inetsim/ftp/ftproot/sample.txt

–$ find / -name 'sample.txt' &> all.txt

This command will put everything in the all.txt file

/dev/null -> is the directory where we delete the files and directories. If we send anything to this directory then it will delete and not show anything int his directory.

$ find / -name 'sample.txt' > location.txt 2>/dev/null

Pipes –

Using pipes, we can connect std output of one command to the std input to another command.

$ ls -a etc/ | less

➔ Here sending ls commands output to the less command.

└─$ ls -a

. all.txt dir_list.txt file file1_soft.txt file3.txt 'file space.txt' my_dir1 space.txt

.. combine1.txt errors.txt file1_hard.txt file2.txt 'file space.txt' location.txt my_dir2 zip__backup.zip

─$ ls -a| head -n 5 | tail -n 2

combine1.txt

dir_list.txt

─$ find / -name 'sample.txt' |& all.txt -> it will show error also

─$ find / -name 'sample.txt' | all.txt -> it'll show paths without errors.

Command History

$ history

➔ It will show the list of commands executed previously with a number.
➔ We can execute the command with this number also.
➔ Size to store/display the commands in history command can be change in config file.

$ history

1 printenv

2 clear

3 COUNT_LOCAL=55

4 echo $COUNT_LOCAL

5 export COUNT_GLOBAL=23

```
 6  echo $COUNT_GLOBAL
 7  unset COUNT_GLOBAL
 8  echo $COUNT_GLOBAL
 9  unset COUNT_LOCAL
10  echo $COUNT_LOCAL
11  clear
12  whois tryhackme.com
13  clear
14  ls
15  ls > dir_list.txt
16  ls
17  cat dir_list.txt
18  clear
19  find -\name 'sample.txt
20  find -\name 'sample.txt'
21  find / -name 'sample.txt'
22  find / -name 'sample.txt' 2> errors.txt
23  find / -name 'sample.txt' &> all.txt
24  find / -name 'sample.txt' &> location.txt>/dev/null
25  find / -name 'sample.txt' &> location.txt 2>/dev/null
26  find / -name 'sample.txt' > location.txt 2>/dev/null
27  clear
28  ls -a | less
29  ls -a| head -n 5 | tail -n 2
30  ls -a
31  history
```

Execute the command with history number.

$!30

It will run the '30 ls -a' command.

With negative number

$ -3 → It will run the 'ls -a| head -n 5 | tail -n 2' command.

$!! → this is for the previous command

Also use up and down arrow keys for the previous commands.

Run Last cat command

$!cat

Command Substitution –

Using this, we can replace a command with its output before the entire command is executed by the shell.

$ ls -a | cat list-dir.txt -> it will still show all the files.

So use ` ` symbol

$ ls -a `cat list-dir.txt` or $ ls -a $(cat list-dir.txt) -> offers same.

- ➔ It tell the shell to execute ` ` this part first and once it is done then it substitute the contents of the file with the command, when then send to the ls command as command line argument.