MANAGIING USER ENVIRONMENT VARIABLES

To get the most from your Linux hacking system, you need to understand environment variables and be adept at managing them for optimal performance, convenience, and even stealth. Technically, there are two types of variables: shell and environment. Environment variables are systemwide variables built into your system and interface that control the way your system looks, acts, and "feels" to the user, and they are inherited by any child shells or processes. Shell variables, on the other hand, are typically listed in lowercase and are only valid in the shell they are set in. In Kali Linux, your environment is your bash shell. Each user, including root, has a default set of environment variables that determine how the system looks, acts, and feels. You can change the values for these variables to make your system work more efficiently, tailor your work environment to best meet your individual needs, and potentially cover your tracks if you need to.

VIEWING AND MODIFYING ENVIRONMENT VARIABLES:

We can view all our default environment variables using env in the terminal.

```
Kali> env
```

Env variables are always uppercase, as in HOME, PATH, SHELL and so on.

To view all environment variables, including shell and local variables, shell functions such as user-defined variables and command aliases:

```
Kali> set | more
```

HISTSIZE: This variable contains the maximum number of commands our command history fil will store.

```
Kali> set | grep HISTSIZE
```

```
HISTSIZE=1000 (by default)
```

This indicates that the terminal will store your last 1,000 commands by default.

CHANGING VARIABLE VALUES FOR A SESSION:

```
Kail> HISTSIZE=0
```

Now, when we try to use the up and down arrow keys to recall our commands, nothing happens because the system no longer stores them. This is stealthy, although it can be inconvenient.

MAKING VARIABLE VALUE CHANGES PERMANENT:

When you change an environment variable, that change only occurs in that particular environment; in this case, that environment is the bash shell session. This means that when you close the terminal, any changes you made are lost, with values set back to their defaults. If you want to make the changes permanent, you need to use the ***export*** command. This command will export the new value from your current environment (the bash shell) to the rest of the system, making it available in every environment until you change and export it again. Variables are strings, so if you run on the cautious side, it isn't a bad idea to save the contents of a variable to a text file before you modify it. For example, since we're about to change the ***PS1*** variable, which controls the information you display in the prompt, first run the following command to save the existing values to a text file in the current user's home directory

```
Kali> echo $HISTSIZE > ~/valueofHIST.txt
```

This way, you can always undo your changes. If you want to be even more cautious and create a text file with all the current settings, you can save the output of the set command to a text file with a command like this one:

```
Kali> set > ~/valueofvars.txt
```

Now let's change the value permanently:

```
Kali> HISTSIZE=0
```

```
Kali> export HISTSIZE
```

Or

```
Kali> export HISTSIZE=0
```

CHANGING SHELL PROMPT:

```
Kail> export PS1="Best Hacking Machine:>>> "
```

It will change the prompt to

```
Best Hacking Machine:>>>
```

The PS1variable has a set of placeholders for information we want to display in the prompt, including the following:

\u the name of the current user

\h the hostname

\W The base name of the current working directory

CHANGING THE PATH:

One of the most important variables in your environment is your PATH variable, which controls where on your system your shell will look for commands you enter, such as cd, ls, and echo. Most commands are located in the ***sbin*** or ***bin*** subdirectory, like ***/usr/local/sbin or usr/local/bin***. If the bash shell doesn't find the command in one of the directories in your PATH variable, it will return the error command not found, even if that command does exist in a directory not in your PATH.

You can find out which directories are stored in your PATH variable by using echo on its contents, like so:

```
Kali> echo $PATH
```

```
>>> /usr/local/sbin:usr/local/bin:/usr/sbin:/sbin/bin
```

These are the directories where your terminal will search for any command. When you enter ls, for example, the system knows to look in each of these directories for the ls command, and when it finds ls, the system executes it. Each directory is separated by a colon (:), and don't forget to add the $content symbol to PATH.

ADDING TO PATH VARIABLE:

You can probably see why it's important to know what is in your PATH variable: if you downloaded and installed a new tool—let's say newhackingtool—into the /root/newhackingtool directory, you could only use commands from that tool when you're in that directory because that directory is not in the PATH variable. Every time you wanted to use that tool, you would first have to navigate to /root/newhackingtool, which is a bit inconvenient if you want to use the tool often. To be able to use this new tool from any directory, you need to add the directory holding this tool to your PATH variable. To add newhackingtool to your PATHvariable, enter the following:

```
Kali> PATH=$PATH:/root/newhackingtool

Kali> echo $PATH

>>> /usr/local/sbin:usr/local/bin:/usr/sbin:/sbin/bin:/root/newhackingtool
```

If that directory is not yet part of our path, we can add it by putting

```
PATH=$HOME/bin:$PATH or PATH=$HOME/any_directory:$PATH
```

in our .bashrc file.

To unset or to delete a variable we can use the unset command.

```
Kali> myVar=10

Kali> echo $myVar          // 10

Kali> unset myVar
```