

Bash Scripting

💡 Scripting is a valuable part of any sysadmin's toolkit. You have been exposed to bash during both your dhcp02 and web01 labs, so this module will focus directly on the use of **B**ourne **A**gain **S**hell, either *interactively* (engages you, the user, for input), or through the use of a script (run and done).

You will need fw02, ad02, wks02 and web01.

SSH into your Linux server from wks01 as a named Linux user (as opposed to a domain user), elevate to root, and then determine your bash version and where the actual bash program resides. Consider downloading PuTTY if you are having formatting issues with ssh in Powershell.

```
root@web01-rubeus:~
PS C:\Users\rubeus-adm> Write-Host ; Write-Host "Who is logged in atm?" ; whoami ; Write-Host ; echo "Which OS is this on atm?" ; hostname ; Write-Host

Who is logged in atm?
rubeus\rubeus-adm

Which OS is this on atm?
ad02-rubeus

PS C:\Users\rubeus-adm> ssh rubeus@web01-rubeus
rubeus@web01-rubeus's password:
Last login: Sat Oct 30 12:47:46 2021 from ad02-rubeus.rubeus.local
Last login: Sat Oct 30 12:47:46 2021 from ad02-rubeus.rubeus.local
[rubeus@web01-rubeus ~]$ sudo -i
[sudo] password for rubeus:
[rubeus@web01-rubeus ~]# echo ; echo "What is the version of BASH atm?" ; bash -version | grep version ; echo ; echo "Where is BASH's file path?" ; which bash ; echo

What is the version of BASH atm?
GNU bash, version 4.2.46(2)-release (x86_64-redhat-linux-gnu)
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

Where is BASH's file path?
/bin/bash

[rubeus@web01-rubeus ~]#
```

The Path Environment Variable is very important. It tells your Bash interpreter, which directories to scan for applications that match your command.

```
root@web01-rubeus:~
[rubeus@web01-rubeus ~]# echo $PATH
/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
[rubeus@web01-rubeus ~]#
```

To show all [Environment Variables](#), enter `env`.

```
root@web01-rubeus:~  
[root@web01-rubeus ~]# env  
XDG_SESSION_ID=1041  
HOSTNAME=web01-rubeus  
SHELL=/bin/bash  
TERM=xterm-256color  
HISTSIZE=1000  
USER=root  
LS_COLORS=rs=0:di=38;5;27:ln=38;5;5  
=48;5;232:38;5;9:mi=05;48;5;232:38;  
10;38;5;21:st=48;5;21;38;5;15:ex=38  
;5;9:*.lzh=38;5;9:*.lzm=38;5;9:*.t  
5;9:*.gz=38;5;9:*.lrz=38;5;9:*.lz=3  
;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=  
8;5;9:*.cpio=38;5;9:*.7z=38;5;9:*.r  
;*.pgm=38;5;13:*.ppm=38;5;13:*.tga=  
;*.svgz=38;5;13:*.mng=38;5;13:*.pcx  
13:*.ogm=38;5;13:*.mp4=38;5;13:*.m4  
3:*.rm=38;5;13:*.rmvb=38;5;13:*.flc  
.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38  
au=38;5;45:*.flac=38;5;45:*.mid=38;  
av=38;5;45:*.axa=38;5;45:*.oga=38;5  
SUDO_USER=rubeus  
SUDO_UID=1001  
USERNAME=root  
PATH=/usr/local/sbin:/sbin:/bin:/us  
MAIL=/var/spool/mail/root  
PWD=/root  
LANG=en_US.UTF-8  
HISTCONTROL=ignoredups  
SHLVL=1  
SUDO_COMMAND=/bin/bash  
HOME=/root  
LOGNAME=root  
LESSOPEN=||/usr/bin/lesspipe.sh %s  
SUDO_GID=1001  
_=/bin/env  
[root@web01-rubeus ~]#
```

Go back to the normal user with `exit`, and check out your path.

```
rubeus@web01-rubeus:~  
[rubeus@web01-rubeus ~]$ echo $PATH  
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/rubeus/.local/bin:/home/rubeus/bin  
[rubeus@web01-rubeus ~]$
```

What has changed? And where does this change come from? Your profile information comes from several files in `/etc`, as well as your home directory.

If you want to modify the path for *all* users, then you would do so in a file in /etc. If you are just changing the user's *specific* environment, then you would do it in configurations located in their home directory.

https://www.gnu.org/software/bash/manual/html_node/Bash-Startup-Files.html

The hidden files .bash_profile and .bashrc are the user specific configuration files in the user's home directory.

💡 Unlike Windows, Linux hidden files are those files that have a leading period ("."). To see these files, enter `ls -la`

Shortcuts

Tab Completion. From your home directory, navigate to /usr/share/firewalld/tests/ just using the minimal number of characters and the tab key to complete. Once there, enter `cd -` to go back to the last directory you were in.

➤ rubeus@web01-rubeus:~

```
rubeus@web01-rubeus ~]$ cd /usr/share/firewalld/
rubeus@web01-rubeus firewallld]$ pwd
/usr/share/firewalld
rubeus@web01-rubeus firewallld]$ cd -
/home/rubeus
rubeus@web01-rubeus ~]$ pwd
/home/rubeus
rubeus@web01-rubeus ~]$
```

Up and Down Arrows

Hit the up arrow until you get back to your original `cd` statement

History

Type `history` to see what has gone on before. If you echo the `$HISTSIZE` environment variable, it shows you how many entries are saved.

💣 This saved history can be a security vulnerability if clear text passwords or other sensitive material is typed and saved as command line parameters. It is also a juicy artifact of interest for digital forensics professionals.

Note: Things like MySQL, Python, and some other services store some info in .history files. It's best to clear them regularly on a production server, or `chmod` them in a way they are not readable by all users/groups.

```

> rubeus@web01-rubeus:~
rubeus@web01-rubeus ~]$ history
 1 sudo hostnamectl set-hostname web01-rubeus
 2 exit
 3 ls /etc/passwd
 4 ls /etc/shadow
 5 clear
 6 sudo hostnamectl set-hostname web01-rubeus
 7 exit
 8 su -i
 9 sudo -i
10 exit
11 nslookup 10.0.5.4
12 sudo -i
13 exit
14 nslookup 10.0.5.4 | grep name

```

The following shows a simple bash script. Use either nano or vi to author this.

```

> hermione@web01-hermione:~
GNU nano 2.3.1 File: info.sh

#!/bin/bash
echo "Welcome to SYS255"
echo "Kernel Version"
uname -a
echo "Linux Version"
cat /etc/redhat-release
echo "Currently Logged in Users"
W

```

You can explicitly invoke bash, and pass your new script as a file parameter.

```

> rubeus@web01-rubeus:~
[rubeus@web01-rubeus ~]$ bash info.sh
Welcome to SYS255
Kernel Version
Linux web01-rubeus 3.10.0-1160.31.1.el7.x86_64 #1 SMP Thu Jun 10 13:32:12 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
Linux Version
CentOS Linux release 7.9.2009 (Core)
Currently Logged in Users
13:08:32 up 6 days, 2:06, 1 user, load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@      IDLE   JCPU   PCPU   WHAT
rubeus    pts/0    ad02-rubeus.rube 12:50      0.00s  0.13s  0.00s  bash info.sh
[rubeus@web01-rubeus ~]$

```

Or, the most common thing to do is to change the file permissions to executable for the current user and leverage the shebang line. Note: in order to execute a file, the file must have the executable flag set. The call to the script must be prepended with `./` in order to execute in the current directory. If there is another test.sh in the path, then it could be invoked with just test.sh.

```

rubeus@web01-rubeus:~
[rubeus@web01-rubeus ~]$ chmod +x info.sh
[rubeus@web01-rubeus ~]$ ./info.sh
Welcome to SYS255
Kernel Version
Linux web01-rubeus 3.10.0-1160.31.1.el7.x86_64 #1 SMP Thu Jun 10 13:32:12 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
Linux Version
CentOS Linux release 7.9.2009 (Core)
Currently Logged in Users
13:10:01 up 6 days, 2:07, 1 user, load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@      IDLE   JCPU   PCPU WHAT
rubeus    pts/0    ad02-rubeus.rube 12:50      1.00s  0.13s  0.00s /bin/bash ./info.sh
[rubeus@web01-rubeus ~]$

```

Example of changing the permissions of a script file:

```

rubeus@web01-rubeus:~
[rubeus@web01-rubeus ~]$ touch TestScript.sh ; ls -la TestScript.sh
-rw-rw-r--. 1 rubeus rubeus 0 Oct 30 13:12 TestScript.sh
[rubeus@web01-rubeus ~]$ chmod 744 TestScript.sh ; ls -la TestScript.sh
-rwxr--r--. 1 rubeus rubeus 0 Oct 30 13:12 TestScript.sh
[rubeus@web01-rubeus ~]$ ./TestScript.sh

```

A parsing script.

We are going to work with the /etc/group and /etc/passwd files. We will run through the example using /etc/group, and you will extend the example to do similar things with the /etc/passwd file.

```

rubeus@web01-rubeus:~
[rubeus@web01-rubeus ~]$ ls -l /etc/passwd /etc/group
-rw-r--r--. 1 root root 687 Oct 24 12:32 /etc/group
-rw-r--r--. 1 root root 1398 Oct 24 12:32 /etc/passwd
[rubeus@web01-rubeus ~]$

```

This one line command (i.e. “one-liner”) will parse the /etc/group file and pluck out the first, third and fourth fields as shown below using [awk](#):

```

rubeus@web01-rubeus:~
[rubeus@web01-rubeus ~]$ awk -F '[:]' '{ print "group:" $1, " groupid:" $3 " members:" $4 }' /etc/group
group:root groupid:0 members:
group:bin groupid:1 members:
group:daemon groupid:2 members:
group:sys groupid:3 members:
group:adm groupid:4 members:
group:tty groupid:5 members:
group:disk groupid:6 members:
group:lp groupid:7 members:
group:mem groupid:8 members:
group:kmem groupid:9 members:
group:wheel groupid:10 members:champuser,rubeus

```

Pipelining with |

In many cases, we wish to filter the results of a script or command down using `grep`. In this case, we only want to show entries with the group "wheel", this should show your sudo users.

Deliverable 1. Provide a screenshot similar to the one below:

```
rubeus@web01-rubeus:~  
rubeus@web01-rubeus ~]$ awk -F '[:]' '{ print "group:" $1, " groupid:" $3 " members:" $4 }' /etc/group | grep wheel  
group:wheel groupid:10 members:champuser,rubeus  
rubeus@web01-rubeus ~]$
```

Take a look at the following documentation:

<http://linux.die.net/man/5/passwd>

/etc/passwd

Your job is to create a similar script to the one that parsed `/etc/group`. We are interested in the name, uid, gid, directory and shell fields. Your output should look similar to this:

```
rubeus@web01-rubeus:~  
rubeus@web01-rubeus ~]$ awk -F '[:]' '{ print "name:" $1, " uid:" $3, " group_id:" $4, " homedir:" $6, " shell:" $7 }' /etc/passwd  
name:root uid:0 group_id:0 homedir:/root shell:/bin/bash  
name:bin uid:1 group_id:1 homedir:/bin shell:/sbin/nologin  
name:daemon uid:2 group_id:2 homedir:/sbin shell:/sbin/nologin  
name:adm uid:3 group_id:4 homedir:/var/adm shell:/sbin/nologin  
name:lp uid:4 group_id:7 homedir:/var/spool/lpd shell:/sbin/nologin  
name:sync uid:5 group_id:0 homedir:/sbin shell:/bin/sync  
name:shutdown uid:6 group_id:0 homedir:/sbin shell:/sbin/shutdown  
name:halt uid:7 group_id:0 homedir:/sbin shell:/sbin/halt  
name:mail uid:8 group_id:12 homedir:/var/spool/mail shell:/sbin/nologin  
name:operator uid:11 group_id:0 homedir:/root shell:/sbin/nologin  
name:games uid:12 group_id:100 homedir:/usr/games shell:/sbin/nologin  
name:ftp uid:14 group_id:50 homedir:/var/ftp shell:/sbin/nologin  
name:nobody uid:99 group_id:99 homedir:/ shell:/sbin/nologin  
name:systemd-network uid:192 group_id:192 homedir:/ shell:/sbin/nologin  
name:dbus uid:81 group_id:81 homedir:/ shell:/sbin/nologin  
name:polkitd uid:999 group_id:998 homedir:/ shell:/sbin/nologin  
name:libstoragemgmt uid:998 group_id:997 homedir:/var/run/lsm shell:/sbin/nologin  
name:abrt uid:173 group_id:173 homedir:/etc/abrt shell:/sbin/nologin  
name:rpc uid:32 group_id:32 homedir:/var/lib/rpcbind shell:/sbin/nologin  
name:sshd uid:74 group_id:74 homedir:/var/empty/sshd shell:/sbin/nologin  
name:postfix uid:89 group_id:89 homedir:/var/spool/postfix shell:/sbin/nologin  
name:chrony uid:997 group_id:995 homedir:/var/lib/chrony shell:/sbin/nologin  
name:ntp uid:38 group_id:38 homedir:/etc/ntp shell:/sbin/nologin  
name:tcpdump uid:72 group_id:72 homedir:/ shell:/sbin/nologin  
name:champuser uid:1000 group_id:1000 homedir:/home/champuser shell:/bin/bash  
name:rubeus uid:1001 group_id:1001 homedir:/home/rubeus shell:/bin/bash  
name:apache uid:48 group_id:48 homedir:/usr/share/httpd shell:/sbin/nologin  
name:tss uid:59 group_id:59 homedir:/dev/null shell:/sbin/nologin  
name:sssd uid:996 group_id:993 homedir:/ shell:/sbin/nologin  
rubeus@web01-rubeus ~]$
```

Deliverable 2. Provide the screenshot running a one liner and its output that you used to produce the similar output above.

Brace expansion

Go ahead and figure out how to install the **tree** package using yum if you haven't already done so (Similar to how we installed DHCP and Apache on Linux).

The following example shows how curly braces { } can be used in common commands to execute multiple commands at the same time.

```
rubeus@web01-rubeus:~  
[rubeus@web01-rubeus ~]$ mkdir -p bashstuff/{dira,dirb,dirc}/sub1/sub2  
[rubeus@web01-rubeus ~]$ tree bashstuff/  
bashstuff/  
├── dira  
│   ├── sub1  
│   │   └── sub2  
├── dirb  
│   ├── sub1  
│   │   └── sub2  
└── dirc  
    ├── sub1  
    │   └── sub2  
  
9 directories, 0 files  
[rubeus@web01-rubeus ~]$
```

Loops


```
rubeus@web01-rubeus:~  
[rubeus@web01-rubeus ~]$ seq 1 10  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
[rubeus@web01-rubeus ~]$ for i in $(seq 1 10); do echo num:$i; done  
num:1  
num:2  
num:3  
num:4  
num:5  
num:6  
num:7  
num:8  
num:9  
num:10  
[rubeus@web01-rubeus ~]$
```

Convert to a script called loop.sh

Note: the semicolons in the one liner are replaced by newlines in the script.

```
rubeus@web01-rubeus:~  
[rubeus@web01-rubeus ~]$ cat loop.sh  
#!/bin/bash  
for i in $(seq 1 10)  
do  
    echo num:$i  
done  
  
[rubeus@web01-rubeus ~]$ bash loop.sh  
num:1  
num:2  
num:3  
num:4  
num:5  
num:6  
num:7  
num:8  
num:9  
num:10  
[rubeus@web01-rubeus ~]$
```


Deliverable 3. Ping Sweeper. Convert the script above, using both the echo and possibly the ping command on the following line (1 ping only). Attempt to ping 192.168.4.1-10. Provide a screenshot showing your updated bash script **syntax**, and its **output**. It should have output similar to that shown below. For a challenge, filter out the **failed** pings.

 Select rubeus@web01-rubeus:~

```
[rubeus@web01-rubeus ~]$ bash pingsweeper.sh
PING 192.168.4.1 (192.168.4.1) 56(84) bytes of data.
From 10.0.17.2 icmp_seq=1 Destination Host Unreachable

--- 192.168.4.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

PING 192.168.4.2 (192.168.4.2) 56(84) bytes of data.
From 10.0.17.2 icmp_seq=1 Destination Host Unreachable

--- 192.168.4.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

PING 192.168.4.3 (192.168.4.3) 56(84) bytes of data.
From 10.0.17.2 icmp_seq=1 Destination Host Unreachable


--- 192.168.4.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

PING 192.168.4.4 (192.168.4.4) 56(84) bytes of data.
64 bytes from 192.168.4.4: icmp_seq=1 ttl=126 time=1.71 ms

--- 192.168.4.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.710/1.710/1.710/0.000 ms
PING 192.168.4.5 (192.168.4.5) 56(84) bytes of data.
64 bytes from 192.168.4.5: icmp_seq=1 ttl=126 time=1.20 ms
```

Deliverable 4. Create an nslookup script (nslu.sh) that provides just the DNS names for those systems found. Use your Virtual LAN address space this time 10.0.5.x. Provide a screenshot showing your updated bash script **syntax**, and your **output** should look similar to the figure below.

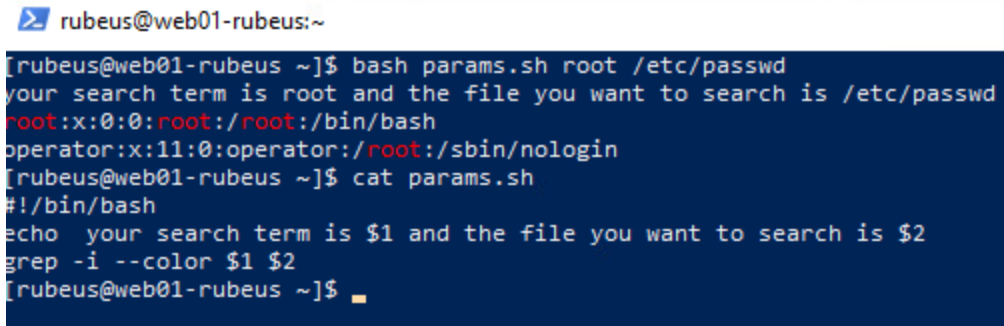
hint: http://ldp.org/LDP/Bash-Beginners-Guide/html/sect_08_02.html

 rubeus@web01-rubeus:~

```
[rubeus@web01-rubeus ~]$ bash nslookup.sh
2.5.0.10.in-addr.arpa  name = fw02-rubeus.rubeus.local.
4.5.0.10.in-addr.arpa  name = web01-rubeus.rubeus.local.
6.5.0.10.in-addr.arpa  name = ad02-rubeus.rubeus.local.
8.5.0.10.in-addr.arpa  name = fs01-rubeus.rubeus.local.
[rubeus@web01-rubeus ~]$
```

Basic input Parameters

Take a look at the following params.sh file. \$1 and \$2 input variables map to the search term and file input to this script.



```
rubeus@web01-rubeus:~  
[rubeus@web01-rubeus ~]$ bash params.sh root /etc/passwd  
your search term is root and the file you want to search is /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
operator:x:11:0:operator:/root:/sbin/nologin  
[rubeus@web01-rubeus ~]$ cat params.sh  
#!/bin/bash  
echo your search term is $1 and the file you want to search is $2  
grep -i --color $1 $2  
[rubeus@web01-rubeus ~]$
```

Deliverable 5. Modify one of your previous scripts to take an input parameter (perhaps a network prefix). Provide a screenshot of both the output and the shell script syntax.

Deliverable 6: Install nmap and create a bash script that will ask for user input on nmap parameters (hint: look up command switches for nmap parameters), and then execute those parameters after nmap is installed. Run an nmap quick scan against your **10.0.5.0/24** network. Provide a screenshot of your script output, as well as the script syntax.