
EIT
TRAINING

TRAINING PLAN

- **INTRODUCTION TO LINUX**
- FORENSICS AND OSINT
- CRYPTOGRAPHY
- WEB EXPLOITATION
- REVERSE ENGINEERING
- BINARY EXPLOITATION

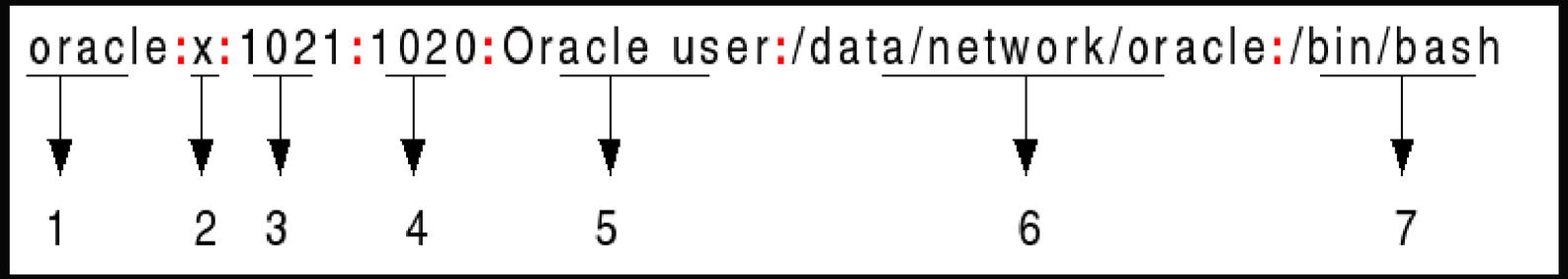
LINUX for CTFs

INTRODUCTION TO PERMISSIONS AND BASH SCRIPTING

Made by :
JEBBARI BADR
SADIK AYMANE

/etc/passwd

users



1 - Username
2 - Password (unused)
3 - UID (User ID)
4 - GID (Group ID)
5 - Comment
6 - Home Directory
7 - Command/Shell

/etc/group

groups

```
cdrom:x:24:vivek,student13,raj
_____ _ __ ____________________
|     |  |          |
|     |  |          |
|     |  |          |
1     2  3          4
```

1 - Group Name
2 - Password (unused)
3 - GID (Group ID)
4 - Members of the group (users)

/etc/shadow

passwords

1 - Username

2 - Hashed Password

3 - Last password change

4 - Minimum

5 - Maximum

6 - Number of days before the expiration

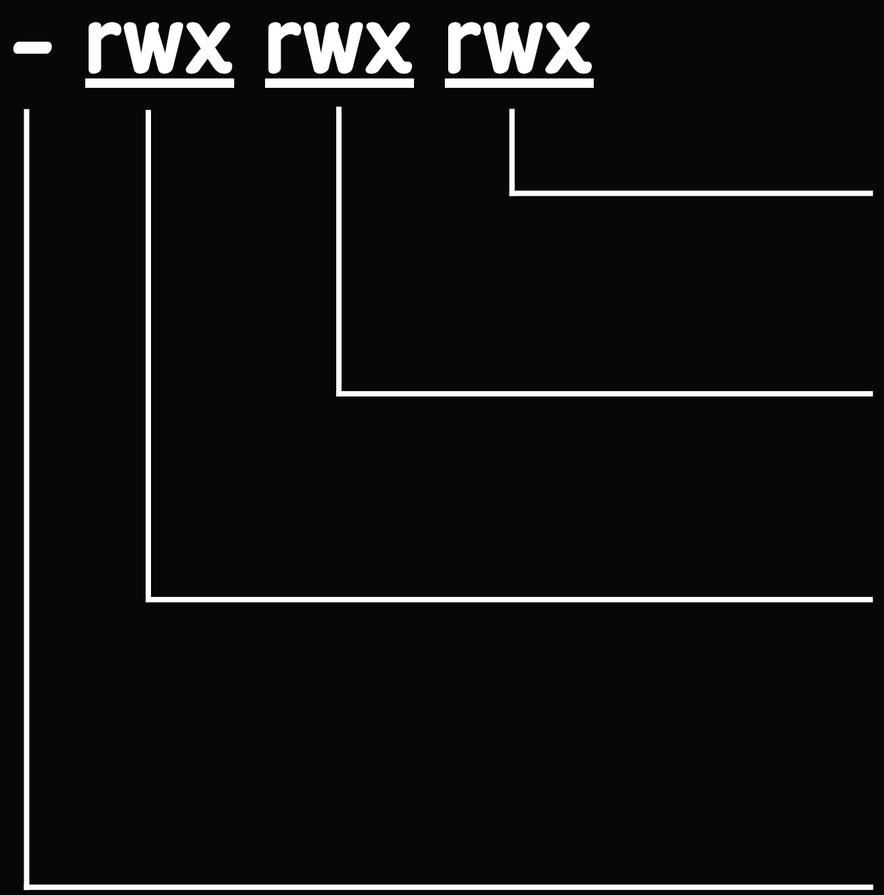
#chown/chgrp

change owner/group

```
┌──(root㉿kali)-[/home/kali/CIT A7SAN CLUB]
└─# ls -l
total 0
-rw-r--r-- 1 kali kali 0 Oct 17 15:50 TEST

┌──(root㉿kali)-[/home/kali/CIT A7SAN CLUB]
└─# chown user1 TEST

┌──(root㉿kali)-[~kali/CIT A7SAN CLUB]
└─# ls -l
total 0
-rw-r--r-- 1 user1 kali 0 Oct 17 15:50 TEST
```

```
┌──(root㉿kali)-[~kali/CIT A7SAN CLUB]
└─# ls -l
total 0
-rw-r--r-- 1 kali kali 0 Oct 17 15:50 TEST

┌──(root㉿kali)-[~kali/CIT A7SAN CLUB]
└─# chgrp group1 TEST

┌──(root㉿kali)-[~kali/CIT A7SAN CLUB]
└─# ls -l
total 0
-rw-r--r-- 1 kali group1 0 Oct 17 15:50 TEST
```

file permissions in linux



Read, write, and execute permissions for all other users.

Read, write, and execute permissions for the group owner of the file.

Read, write, and execute permissions for the file owner.

File type:

- - indicates a regular file
- d indicates directory
- l indicates a soft link

#chmod

change permissions

$ chmod u+r g+w o+x somefile

```
-  ---  ---  ---   →   -  r--  -w-  --x
```

$ chmod u-w g-x o-r somefile

```
-  rwx  rwx  rwx   →   -  r-x  rw-  -wx
```

#chmod

change permissions

drwxrwxrwx

d = Directory
r = Read
w = Write
x = Execute

7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

chmod 777

rwx | rwx | rwx
Owner | Group | Others

$ chmod 777 somefile

\- rwx rwx rwx

$ chmod 754 somefile

\- rwx r-x r--

#su

switch user

```
$ su [user]
```

#sudo

do as

```
$ sudo -l                  list user's privileges
$ sudo <cmd>               do command as root
$ sudo -u <user> <cmd>     do command as specified user
```

Introduction to Bash Scripting

Bash Scripting

A Bash script is a plain-text file that contains a series of commands that are executed as if they had been typed at a terminal prompt.

They have the extension .sh and begin with #!/bin/bash.

#variables

Syntax: name=value

```
kali@kali:~$ first_name=CIT
kali@kali:~$ last_name=Security
kali@kali:~$ echo $first_name $last_name
CIT Security
```

There is a difference between single and double quotes.

```
kali@kali:~$ greeting='Hello $first_name'
kali@kali:~$ greeting2="Hello $first_name"
kali@kali:~$ echo $greeting
Hello $first_name
kali@kali:~$ echo $greeting2
Hello CIT
```

#variables

$(<cmd>): set the value of a variable to the output of a command/program

```
kali@kali:~$ me=$(whoami)
kali@kali:~$ echo $me
kali
kali@kali:~$ range=$(seq 1 100)
kali@kali:~$ echo $range
1
2
...
100
```

#numerical operations

((operation))

```
kali@kali:~$ num=2; echo $num
2
kali@kali:~$ num=$((num+15)); echo $num
17
kali@kali:~$ ((num++)); echo $num
18
kali@kali:~$ ((num=num+5)); echo $num
23
kali@kali:~$ echo $((num+3))
26
```

#input

read: take user input

```
kali@kali:~$ cat script.sh

#!/bin/bash
read test
echo "Hello $test"

kali@kali:~$ ./script.sh
CIT
Hello CIT
```

#input

$1 - $9: first 9 arguments passed to the script

```
kali@kali:~$ cat script.sh

#!/bin/bash
echo $2 $3 $4 $5 $1

kali@kali:~$ ./script.sh arg1 arg2 arg3 arg4 arg5 arg6
arg2 arg3 arg4 arg5 arg1
```

#global variables

dear god

Variable Name	Description
$0	The name of the Bash script
$1 - $9	The first 9 arguments to the Bash script
$#	Number of arguments passed to the Bash script
$@	All arguments passed to the Bash script
$?	The exit status of the most recently run process
$$	The process ID of the current script
$USER	The username of the user running the script
$HOSTNAME	The hostname of the machine
$RANDOM	A random number
$LINENO	The current line number in the script

#control flow

if, elif, else

```
if [ <some test> ] && [ <some test> ]
then
    <perform action>
elif [ <some test> ] || [ <some test> ]
then
    <perform different action>
else
    <perform another different action>
fi
```

&&: logical AND
||: logical OR

#testing operators

dear god part 2

Operator	Description: Expression True if...
!EXPRESSION	The EXPRESSION is false.
-n STRING	STRING length is greater than zero
-z STRING	The length of STRING is zero (empty)
STRING1 != STRING2	STRING1 is not equal to STRING2
STRING1 = STRING2	STRING1 is equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is equal to INTEGER2
INTEGER1 -ne INTEGER2	INTEGER1 is not equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is less than INTEGER2
INTEGER1 -ge INTEGER2	INTEGER1 is greater than or equal to INTEGER 2
INTEGER1 -le INTEGER2	INTEGER1 is less than or equal to INTEGER 2
-d FILE	FILE exists and is a directory
-e FILE	FILE exists
-r FILE	FILE exists and has read permission
-s FILE	FILE exists and it is not empty
-w FILE	FILE exists and has write permission
-x FILE	FILE exists and has execute permission

#for loop

```
for var in <list>
do
    <perform action>
done
```

#while loop

```
while [ <some test> ]
do
    <perform action>
done
```

#challenges

have fun c:

- Write a script that prints numbers from 1 to 50.
- Write a script that compares two numbers and gives the result of the comparison.
- Write a script that prints the ip address 10.0.0.x for x between 1 and 100
- Write the same script as before in one line.

Extra challenges when you're done c: