Contents

1 Commands Ubuntu

1.1 Bash

The language bash is a script- language that allow to write any type of script.

Structure

Variable

- **declaretion**: name_vabiable (without any sign before)
- **assegnament**:
  - Value: name_variable='Value'
  - command's result: name_vabiable=$(command)

iteration

- for variable in {}; do .... done ({} can be substituted by range { left_value..right_value } or also set [value1 value2 ... valueN])

Condition

- if [conditiod]: then .... fi

1.2 Connection servers

1.2.1 SSH

Structure **ssh arguments**: Create an openSSH connection

Arguments

- name_user@name_host (can be also localhost) **example** bandit0@bandit.labs.overthewire.org
- -p: used to indicate port's number **example** -p 2220
- -i : to use private password of asymmetric connection (permission key: 400)
- /bin/(shell name): allow to log in with another shell different by bash

1.2.2 NC (NetCat)

Useful to menage net's connections, or also to send data to one specific port.
nc arguments : ex. send password to 30000 port: **echo ¡password¿ — nc ¡ip_address¿ 30000 (port)**

arguments

- destination : ip address destination
- port: destination port
- -l : allow to create a server that attend a connection on a specific port

1.2.3 OPENSSL

It is a toolkit that has many functions, these permit, for example, to create secure connections.
openssl commands option parameters

arguments

- s_clients: protocol

parameters

- host: destination host
- port: destination port

1.2.4 NMAP

Useful tool that allowed to scan port and has also other functions
nmap arguments host

arguments

- -p : select a range of ports ex 1-200
- -sV: scan deeper range of port and find versione and service .

1.2.5 GIT

Allow to use git system. **git option path**

Option

clone Allow to clone repository in local or in another repository or local directory

- ssh-path: ssh://username@hostname[:port]/path_directories

show Show any details about some Git objects as blob, tree, tag e commit

log Allow to see commit history into an git repository.
what we see

Secure Hash Algorithm : unique commit ID

Author: commit Author name

date: date and hour

Commit message: Descriptive message about commit

checkout Allow to switch from branch to other, restore a file in a worktree or overwrite paths with index or commit content.

for restore add : SHA commit **es** git checkout [commit SHA]

switch from branch : path branch **es** git checkout remotes/origin/dev

branch It Shows branches on repository

- **-** a : show hidden branch

tag without option allows to show all tag, instead with option allows to add new tag

add allow to add new modify to next commit

commit it executes commit and select what modify will be sent to remotes branch

push it sent commit from local to remote branch

- source destination : **es** git push origin master

1.3 Simple Commands

1.3.1 CAT

Strange name of file

- If name of a file is a particular symbol, and not a lettere or a number, like a dot or ”-”. You can read it by writing before the name ”./”(it’s for all commands). **example** namefile: - -¿ ”cat ./-” .
- If name of a file contain spaces, you can read it by writing namefile between double quote. **example** namefile: nome con spazi -¿ cat ”nome con spazi” .
- if file is hidden you can read it anyway by writing ”.” befor the neme **example** hidden name file: .hidden -¿ cat .hidden

1.3.2 FILE

Explain what type of file is it. **file NameFilesWithAlsoRegularEXP/ArgumentsAndNameFiles**
example

- file * : allow to show types of all file in the folder
-

1.3.3 FIND

Useful to search a file or a directory.
find path arguments : es. find ./inhere -type f -size 1033c -not -executable -exec find +;q

arguments It can be divided in 2 group: TEST and ACTION, in the first group there are all filter that allowed to find right file. In the second group are listed all action that we can do after the filter are finished. And there are also many operators, these permitted to try more combination of filter.

TESTS Tests that i have tried.

- type : f-¿ file, d-¿directory, b-¿ block (buffered) special ...
- size : [+/-/nothing] number + unit -¿ c: Bytes, k:kibibytes, M:mebibytes, G:gibibytes and overthewire
- executable: select only file that are executable
- group: filter to group user
- user: filter to user

operator

- -not ! : it's used to negate a filter

ACTION Action that i have tried.

- exec: it permit to run other commands that will use output of find as input **es.** -exec file (if i want add arguments of this command i have to write before of curly braces) +; "" will be substitued by output of commands, "+" is a terminator that allowed to append comamnd's results to "find"'s results.

1.3.4 GREP

Useful to search text into file. **grep "word" file**

arguments

- -w : used to search a specif word, select only lines containg mathces that form whole words.

1.3.5 SORT

sort file arguments
Used to sort a file text **es** sort data.txt — uniq -c

1.3.6 UNIQ

uniq file arguments

arguments

- -c: count number of lines that are equals, only if they are under each other

1.3.7 STRINGS

Command to extract strings by file data
strigns file

1.3.8 BASE64

Used to encode/decode a file **base64 arguments file**

arguments

- -d : used to decode a file encoded with base64

1.3.9 TR

used to translate a string, for example to use Rot13 ”tr ’A-Za-z’ ’N-ZA-Mn-za-m’

1.3.10 GZIP

Compress and decompress file, create archivie and other. . . **gzip arguments archivie**: archivie’s name must finish with .gz to extract its.

arguments

- -d : to decompress file.

1.3.11 BZIP2

Compress and decompress file, create archivie and other. . . **bzip2 arguments archivie**

arguments

- -d : to decompress file.

1.3.12 TAR

Compress and decompress file, create archivie and other. . . **tar arguments -f archivie**

arguments

- -x : to extract file.

1.3.13 XXD

To create or to revert an hexdump
xxd arguments file_source file_destination

arguments

- r: hexdump -¿ original

1.3.14 DIFF

Print difference between two file
diff file1 file2

1.3.15 CRON

The cron command is a time-based job scheduler in Unix-like operating systems. It is used to schedule tasks (commands or shell scripts) to run periodically at fixed times, dates, or intervals. `cron` is typically used for automating system maintenance, administrative tasks, or recurring processes.

1.3.16 CRONTAB

The crontab command is used to interact with the cron daemon. It allows users to create, edit, list, and remove cron jobs (scheduled tasks). Each user on a Unix-like system can have their own `crontab` file, which defines the schedule for their individual cron jobs.

Here is an example of a `crontab` entry:

```
# m h   dom mon dow     command
*/15 * * * *    /path/to/script.sh
```

In this example, `/path/to/script.sh` will be executed every 15 minutes.

1.4 Description of Directory

1.4.1 /etc/cron.d

The `/etc/cron.d` directory is used for system-specific cron jobs. Instead of editing a single system-wide `crontab` file, individual cron jobs can be placed in separate files within this directory. The format of these files follows the standard cron job syntax. Each file in `/etc/cron.d` specifies a cron job and the user under which the job should run.

1.4.2 /var/spool

The `/var/spool` directory is used to hold files that are queued for some kind of processing. It typically contains directories related to various services on the system. Here are some common directories found within `/var/spool`:

- `/var/spool/cron`: Contains user-specific cron files (`crontab` files) which are managed by the `crontab` command.
- `/var/spool/mail`: Holds user mailbox files where incoming emails are stored for local delivery.
- `/var/spool/cups`: Stores print jobs in the CUPS (Common Unix Printing System) printing queue.
- `/var/spool/at`: Used by the `at` command to store jobs scheduled for one-time execution.

The `/var/spool` directory itself does not contain subdirectories directly related to cron jobs, but it is crucial for the system's operational workflow, particularly in managing queued and pending tasks like email deliveries and print jobs.

1.4.3 /etc/passwd

The `/etc/passwd` file is a system file found on Unix and Unix-like operating systems. It stores essential user account information that is used during user authentication and login processes. Each line in the `passwd` file represents a user account and contains seven colon-separated fields:

- **Username (`username`):** This is the name of the user account.
- **Password (`password`):** Traditionally, this field used to contain the encrypted password for the user account. However, modern systems often store an 'x' character here, with the actual password hash stored in the `/etc/shadow` file for security reasons.
- **User ID (`UID`):** Each user is assigned a unique numerical User ID (UID). This ID is used by the system to identify the user.
- **Group ID (`GID`):** This field represents the primary group ID associated with the user.
- **User Info (`user information`):** This field typically contains additional information about the user, such as the user's full name, phone number, and other details.
- **Home Directory (`home directory`):** This is the absolute path to the user's home directory, where the user will be placed after logging in.
- **Shell (`login shell`):** This field specifies the default shell or command interpreter for the user when they log in. If this field is empty or set to `/bin/false`, the user will not be allowed to log in.

Here is an example line from the `/etc/passwd` file:

```
john:x:1001:1001:John Smith:/home/john:/bin/bash
```

In this example: - Username: `john` - Password: `x` (actual password hash stored in `/etc/shadow`) - UID: `1001` - GID: `1001` - User Info: `John Smith` - Home Directory: `/home/john` - Shell: `/bin/bash`

It's important to note that direct editing of the `/etc/passwd` file is not recommended. User management tasks should be performed using dedicated commands such as `useradd`, `usermod`, and `userdel`, which manage both the `/etc/passwd` and `/etc/shadow` files securely.

1.5 Manage Process

1.5.1 &

Create a background process with any command
command attribute &

1.6 Programm

1.6.1 Vim

Commands

- : -¿ Mode command line
- :e -¿ open a file by current
- i -¿ insert mode
- :set -¿ set up some option and it can be used to set up an enviromental variable like SHELL **ex** :set shell=/bin/bash

1.7 variable of Linux

1.7.1 $0

It is a variable that contain programm's name in execution.

2 Wireshark

2.1 Details

2.2 Filter

2.2.1 Multifilter

You can write a filter or a multifilter using logic operator as && (AND), ——(OR) and other.

2.2.2 Type of Filter

There are many kind of filter already write:

pkt_comment It can be used to search comment on packet.

frame Allow to analyze directly attribute of the frame

data Allow to analyze data are contained in the pack

3 Python for webservice

3.1 requests

3.1.1 GET

It is a method of the module requests, that cna take as input an URL **Es. request.get(url)**. This permit to obtain a JSON server response, and it can be printed as simple text with method *text*

parameters

url : server address

params : parameters that could be added to url to obtain other information or a specif page**Es. response.get(url, params=paylod), paylod is a dictionary**

headers : it allowed to change headers of response **Es. requests.get(url,headers=variable)**

Accept : it allow to change format that server have to send

cookies : it allow to select any cookies by server

3.1.2 Session()

Create a session that permit to save cookies, it is a method of *requests* and has a own get method

3.1.3 HEAD

It allow to recevie only headers. To print results **response.headers**

3.1.4 POST

To use POST method is like GET method, you must insert an URL, but you can also send complex data

Old way to send data Create a dictionary and put it into a data parameters **requests.post(url,data=data)**, this way is old and few used

JSON New way is JSON, and method POST has a parameters JSON that obbligate who respond to use json format **requests.post(url,json=data)**

3.2 BeautifulSoup

3.2.1 find_all

find_all

3.3 PyCryptodome

3.4 Other Function

3.4.1 Class bytes

method

chr() : convert from decimal to ASCII

fromhex() : return a type bytes from a hexdecimal string **Es. bytestr = bytes.fromhex('666c61677b68337834646563696d616c5f63346e5f62335f41424144424142457d'**

to_bytes(lenght,endianess) : convert an integer in bytes

base64 : library that contain command b64decode() used to decode a string coded in base64

tip

- if you want search in a bytes string, for example with command find(), you must use this method *b' '*

4 SQL Injection

Info