README

Linux Privilege Escalation:

- By Sai Sathvik
- In order to understand what a particular Linux command does, use: https://www.explainshell.com/
- Important Resource: https://null-byte.wonderhowto.com/how-to/crack-shadow-hashes-after-getting-root-linux-system-0186386/
- Privilege escalation is the technique used to escalate our privileges from lower user to higher user(interms of privileges)
- Once we obtain the higher level privilege on the system then we can do a lot of things on that system

Enumeration:

- Here we're going to see few commands which help us in enumerating target system

1. `hostname` - lists the name of the host
2. `uname -a` - prints kernel information
3. `cat /proc/version` - prints almost same infor of above command but more like gcc version....
4. `cat /etc/issue` - exact version on the OS
5. `ps` - lists the processes that are running
    - `ps -A` - all running processes
    - `ps axjf` - process tree
    - `ps aux` - displays processes with the users as well
6. `env` - shows all the environment variable
7. `sudo -l` - lists the commands that any user run as root without password
8. `groups` - lists the groups that current user is in
9. `id` - lists id of group,user
10. `cat /etc/passwd` - displays all the user
    - `cat /etc/passwd | cut -d ":" -f 1` - removes other stuff & only displays users
    - `ls /home` - displays users
11. `history` - previously ran commands which might have some sensitive info
12. `ifconfig` (or) `ip a` (or) `ip route` - network related information
13. **netstat** - network route
    - `netstat -a` - all listening and established connection
    - `netstat -at` - tcp connections
    - `netstat -au` - udp connections
    - `netstat -l` - listening connections
    - `netstat -s` - network statistics
    - `netstat -tp` - connections with service name and pid we can also add "l" for only listening ports
    - `netstat -i` - interface related information

- `netstat -ano`

14. **find** command which helps us in finding lot of stuff,
    - Syntax: `find <path> <options> <regex/name>` find . -name flag1.txt: find the file named "flag1.txt" in the current directory
    - `find /home -name flag1.txt` : find the file names "flag1.txt" in the /home directory
    - `find / -type d -name config` : find the directory named config under "/"
    - `find / -type f -perm 0777` : find files with the 777 permissions (files readable, writable, and executable by all users)
    - `find / -perm a=x` : find executable files
    - `find /home -user frank` : find all files for user "frank" under "/home"
    - `find / -mtime 10` : find files that were modified in the last 10 days
    - `find / -atime 10` : find files that were accessed in the last 10 day
    - `find / -cmin -60` : find files changed within the last hour (60 minutes)
    - `find / -amin -60` : find files accesses within the last hour (60 minutes)
    - `find / -size 50M` : find files with a 50 MB size
    - `find / -writable -type d 2>/dev/null` : Find world-writeable folders
    - `find / -perm -222 -type d 2>/dev/null` : Find world-writeable folders
    - `find / -perm -o w -type d 2>/dev/null` : Find world-writeable folders
    - `find / -perm -o x -type d 2>/dev/null` : Find world-executable folders
    - We can also find programming languages and supported languages: `find / -name perl*`, `find / -name python*`, `find / -name gcc*` ...etc

    - `find / -perm -u=s -type f 2>/dev/null` : Find files with the SUID bit, which allows us to run the file with a higher privilege level than the current user. This is important!
15. We can even make use of "grep", "locate", "sort"...etc

Automated Enumeration Scripts:

- In real life we dont get much time to do enumeration so we can make use of some cool automated scripts like follows,
- LinPeas: https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS
- LinEnum: https://github.com/rebootuser/LinEnum
- LES (Linux Exploit Suggester): https://github.com/mzet-/linux-exploit-suggester
- Linux Smart Enumeration: https://github.com/diego-treitos/linux-smart-enumeration
- Linux Priv Checker: https://github.com/linted/linuxprivchecker

---

Linux Kernel Exploits:

- After finding the version of Kernel simple google for that exploit or you can also use "Linux Exploit suggester"
- Once you find the exploit for the privesc, transfer the payload from your machine to target machine and execute and you're good to go.
- In an example I worked out with **overlayfs** exploit and got higher privileges

---

Sudo:

- This one of the first step to do, when you get access to the machine just simpley run "sudo -l", which lists all the files that we can run as root without any password
- Once you have any to run then navigate to https://gtfobins.github.io/ and search for is the one specified is a system program or else modify the file with "/bin/sh" and run that
- GTFO bins is going to be saviour!

SUID:(Set owner User ID)

- Its a kind of permission which gives specific permissions to run a file as root/owner
- This is really helpful to test.
- `find / -perm -u=s -type f 2>/dev/null` this will list all the suid files
- Then later search in GTFObins and look for the way to bypass
- Resource: https://null-byte.wonderhowto.com/how-to/crack-shadow-hashes-after-getting-root-linux-system-0186386/

Capabilities:

- Capabilities are a bit similar to the SUID
- Capabilities provide a subset of root privileges to a process or a binary
- In order to look for them use `getcap -r / 2>/dev/null`
- Find the binary and check that on **GTFOBins** where there's a function for **Capabilities** and try out those any of them will work!
- In the example they provided a capability for `vim` and I used `./vim -c ':py3 import os; os.setuid(0);`

`os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'` which is provided in the website itself and I got root!

- Remember that this process is hit or trail, if it doesnt work move on!

Cron jobs:

- Crons jobs are used for scheduling! Here we can schedule any binary/process to run.
- Interesting part here is that by default they run with the owner privileges.
- But if we find any cron-job which we can edit then we can do a lot!
- Cron job config is stored as **crontabs**
- To view crontab, `cat /etc/crontab`
- Any one can view it!
- Now we'll can see some cron-jobs see whether you can edit or not, if you can then edit with some reverse shell and listen!

PATH:

- PATH is an environment variable
- In order to run any binary we need to specify the full path also, but if the address of file is specified in PATH variable then we can simpley run the binary by mentioning its name, like how we run some command line tools like ls, cd,....etc
- In order to view the content in PATH variable we need to run `echo $PATH` and the outpur will be something like this `usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin`

- So whenever you use a tool without specifying path it searches in PATH and it runs!
- We can even add new path to PATH variable by `export PATH=<new-path>:$PATH`
- Also we need to find a writable paths so run `find / -writable 2>/dev/null`
- In the example I found a location where there's a script when I run its showing that "thm" not found, also it can be run as ROOT
- So I created a binary like `echo "/bin/bash" > thm` and gave executable rights then later added the path where **thm** located to PATH variable and now when I ran the binary then I got root!

NFS:(Network File Sharing)

- In order to view the configuration of NFS run `cat /etc/exports` or also we can type `showmount -e <target IP>` on our machine to find the **mountable shares**.
- In the output look for directories having `no_root_squash`, this means that the particular share is *writable*, hence we can do something to acquires root!
- Now after getting some directories where we can play around lets navigate to our attacker machine and create a sample directory anywhere like `/tmp` ...etc
- Now we need to mount to the target machine by, `mount -o rw <targetIP>:<share-location> <directory path we created>`, here `rw` means read, write privileges.
- Now go to the folder we created and create a binary which gives us root on running.
- Then go back to the target machine and we can view the binary we created in the place we mounted, now run that and get root privileges!(do note that giving executable

rights is not sufficient, we also need to give share rights by `chmod +s <binary>`)