Name : Kishan Shah

Student Id : 23970203

Course : CISC 7320x - Computer Security

Project : Final Computer Security Project

Professor : Tzipora Halevi

John the Ripper & **Wireshark**

In this final project, I am using 2 software to cover overall whole scenario of packet sniffing. First, I am covering John the Ripper software which is an open source and mainly used for password cracking.

John the Ripper

Hacking is not necessarily criminal, although it can be a tool used for bad intentions. John the Ripper (JtR) is one of the most popular password cracking programs out there.

JtR supports several common encryption technologies out-of-the-box for UNIX and Windows-based systems. JtR autodetects the encryption on the hashed data and compares it against a large plain-text file that contains popular passwords, hashing each password, and then stopping it when it finds a match. Pretty Simple!

A very common use case for JtR is to log into an administrative account.

JtR also includes its own wordlists of common passwords for 20+ languages. These wordlists provide JtR with thousands of possible passwords from which it can generate the corresponding hash values to make a high-value guess of the target password. Since most people choose easy-to-remember passwords, JtR is often very effective even with its out-of-the-box wordlists of passwords.

What is John the Ripper Used for?

JtR is primarily a password cracker used during pentesting exercises that can help IT staff spot weak passwords and poor password policies.

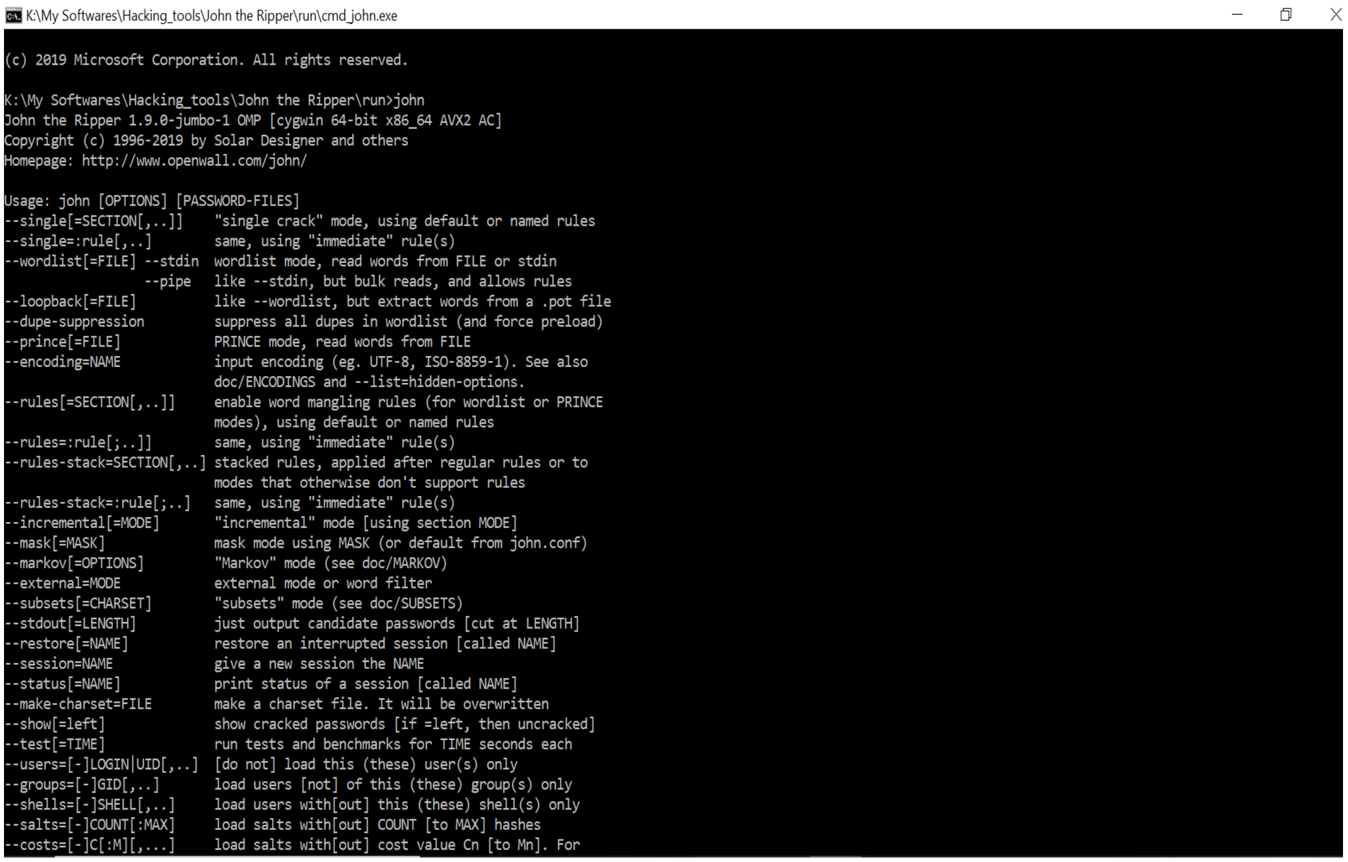
Here is the list of encryption technologies found in JtR:

- UNIX crypt(3)
- Traditional DES-based
- Bigcrypt
- BSDI extended DES-based
- FreeBSD MD5-based (Linux and Cisco IOS)
- OpenBSD Blowfish-based

- Kerberos/AFS
- Windows LM (DES-based)
- DES-based tripcodes
- SHA-crypt hashes (newer versions of Fedora and Ubuntu)
- SHA-crypt and SUNMD5 hashes (Solaris)

To see a list of commands in JtR, run this command:

- ➢ Command : john



K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
(c) 2019 Microsoft Corporation. All rights reserved.

K:\My Softwares\Hacking_tools\John the Ripper\run>john
John the Ripper 1.9.0-jumbo-1 OMP [cygwin 64-bit x86_64 AVX2 AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,..]]    "single crack" mode, using default or named rules
--single=:rule[,..]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin  wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
--loopback[=FILE]          like --wordlist, but extract words from a .pot file
--dupe-suppression         suppress all dupes in wordlist (and force preload)
--prince[=FILE]            PRINCE mode, read words from FILE
--encoding=NAME            input encoding (eg. UTF-8, ISO-8859-1). See also
                           doc/ENCODINGS and --list=hidden-options.
--rules[=SECTION[,..]]     enable word mangling rules (for wordlist or PRINCE
                           modes), using default or named rules
--rules=:rule[;..]]        same, using "immediate" rule(s)
--rules-stack=SECTION[,..] stacked rules, applied after regular rules or to
                           modes that otherwise don't support rules
--rules-stack=:rule[;..]   same, using "immediate" rule(s)
--incremental[=MODE]       "incremental" mode [using section MODE]
--mask[=MASK]              mask mode using MASK (or default from john.conf)
--markov[=OPTIONS]         "Markov" mode (see doc/MARKOV)
--external=MODE            external mode or word filter
--subsets[=CHARSET]        "subsets" mode (see doc/SUBSETS)
--stdout[=LENGTH]          just output candidate passwords [cut at LENGTH]
--restore[=NAME]           restore an interrupted session [called NAME]
--session=NAME             give a new session the NAME
--status[=NAME]            print status of a session [called NAME]
--make-charset=FILE        make a charset file. It will be overwritten
--show[=left]              show cracked passwords [if =left, then uncracked]
--test[=TIME]              run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..]  [do not] load this (these) user(s) only
--groups=[-]GID[,..]       load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]     load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX]     load salts with[out] COUNT [to MAX] hashes
--costs=[-]C[:M][,...]     load salts with[out] cost value Cn [to Mn]. For
```

John the Ripper’s primary modes to crack passwords are single crack mode, wordlist mode, and incremental. The single crack mode is the fastest and best mode if you have a full password file to crack. Wordlist mode compares the hash to a known list of potential password matches. Incremental mode is the most powerful and possibly won’t complete. This is a classic brute force mode that tries every possible character combination until you have a possible result.

The easiest way to try cracking a password is to let JtR go through a series of common cracking modes. If we have a file which contains a hash value of any password then,

- ➢ Command : john password_file_name

This command tells JtR to try “simple” mode, then the default wordlists containing likely passwords, and then “incremental” mode.

We can also download different wordlists from the Internet and can create our own new wordlists for JtR to use with the --wordlist parameter.

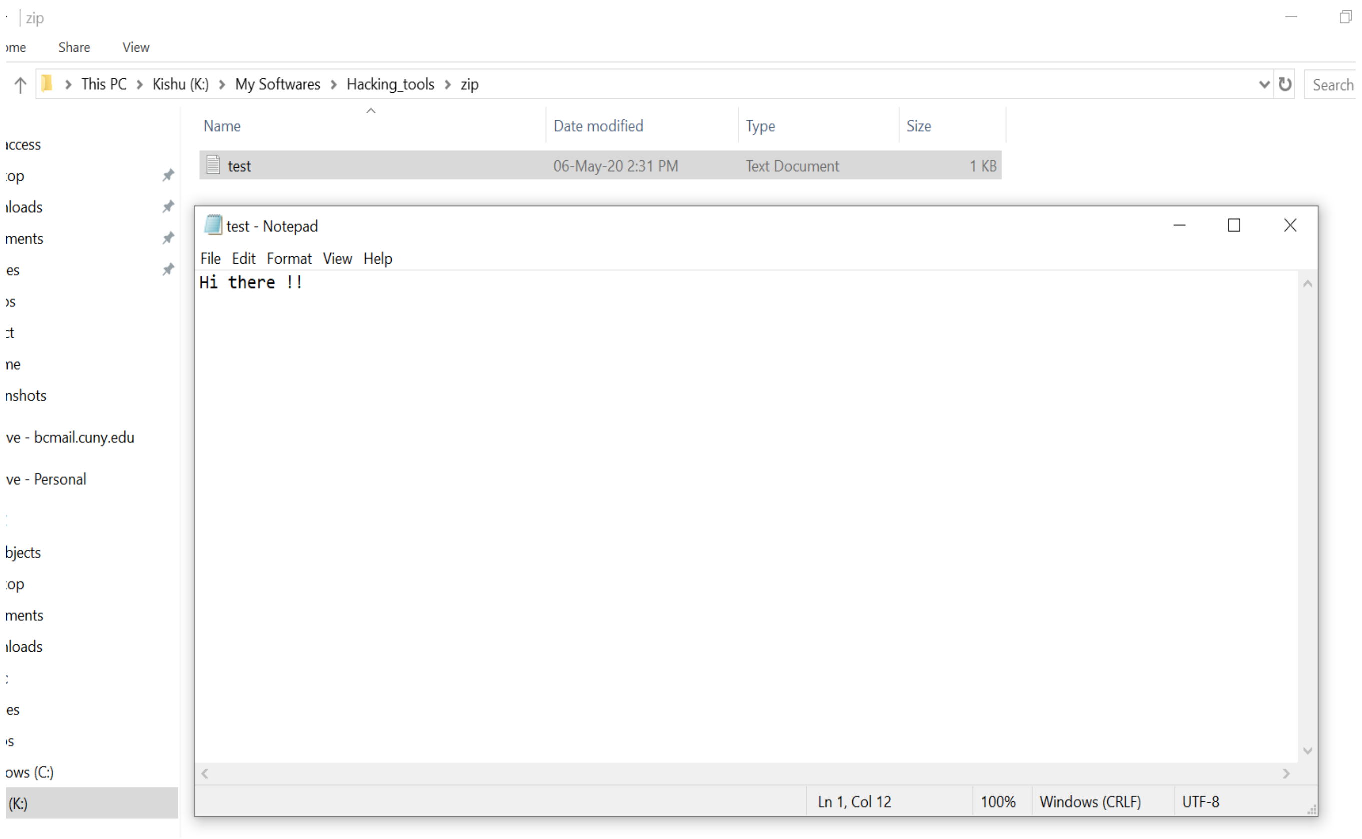
- ➢ Command : john password_file_name -wordlist = “wordlist_file_name.txt”

If we want to specify a cracking mode, use the exact parameter for the mode.

- ➢ Command : john --single password_file_name
- ➢ Command : john --incremental password_file_name

Now, let us see an example. Here I am first zip a text file with password and then I will get hash value of that password from that zip file. Then I will use JtR to get the original password of that file.

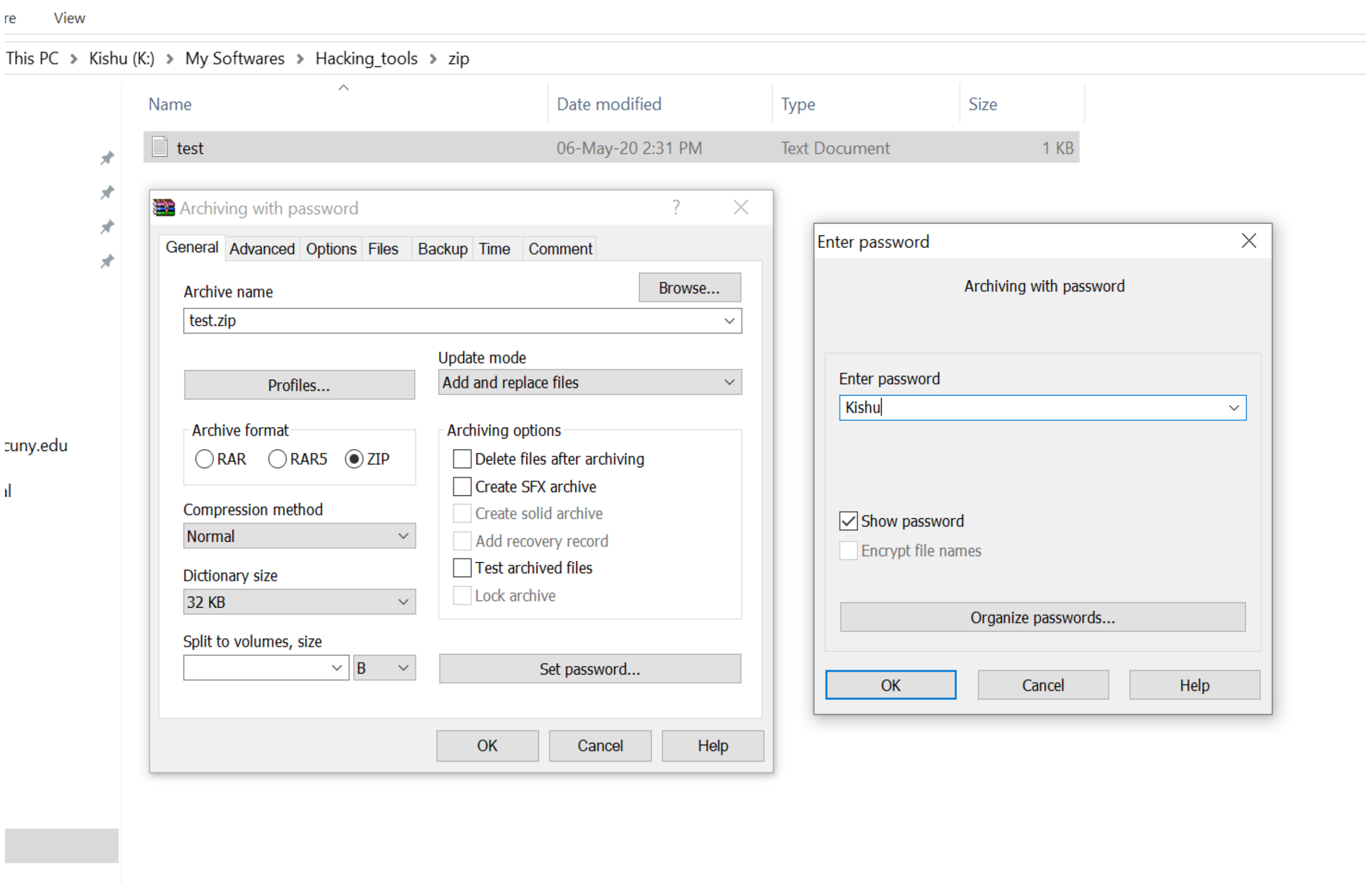
(1) First, I create a text file test.txt



Name	Date modified	Type	Size
test	06-May-20 2:31 PM	Text Document	1 KB

```
Hi there !!
```

(2) Now I zip this text file and set a password for that as well. I am giving password = “Kishu”.



test 06-May-20 2:31 PM Text Document 1 KB

Archiving with password

Archive name: test.zip

Update mode: Add and replace files

Archive format: RAR RAR5 [✓ ZIP]

Compression method: Normal

Dictionary size: 32 KB

Split to volumes, size: B

Archiving options:
- ☐ Delete files after archiving
- ☐ Create SFX archive
- ☐ Create solid archive
- ☐ Add recovery record
- ☐ Test archived files
- ☐ Lock archive

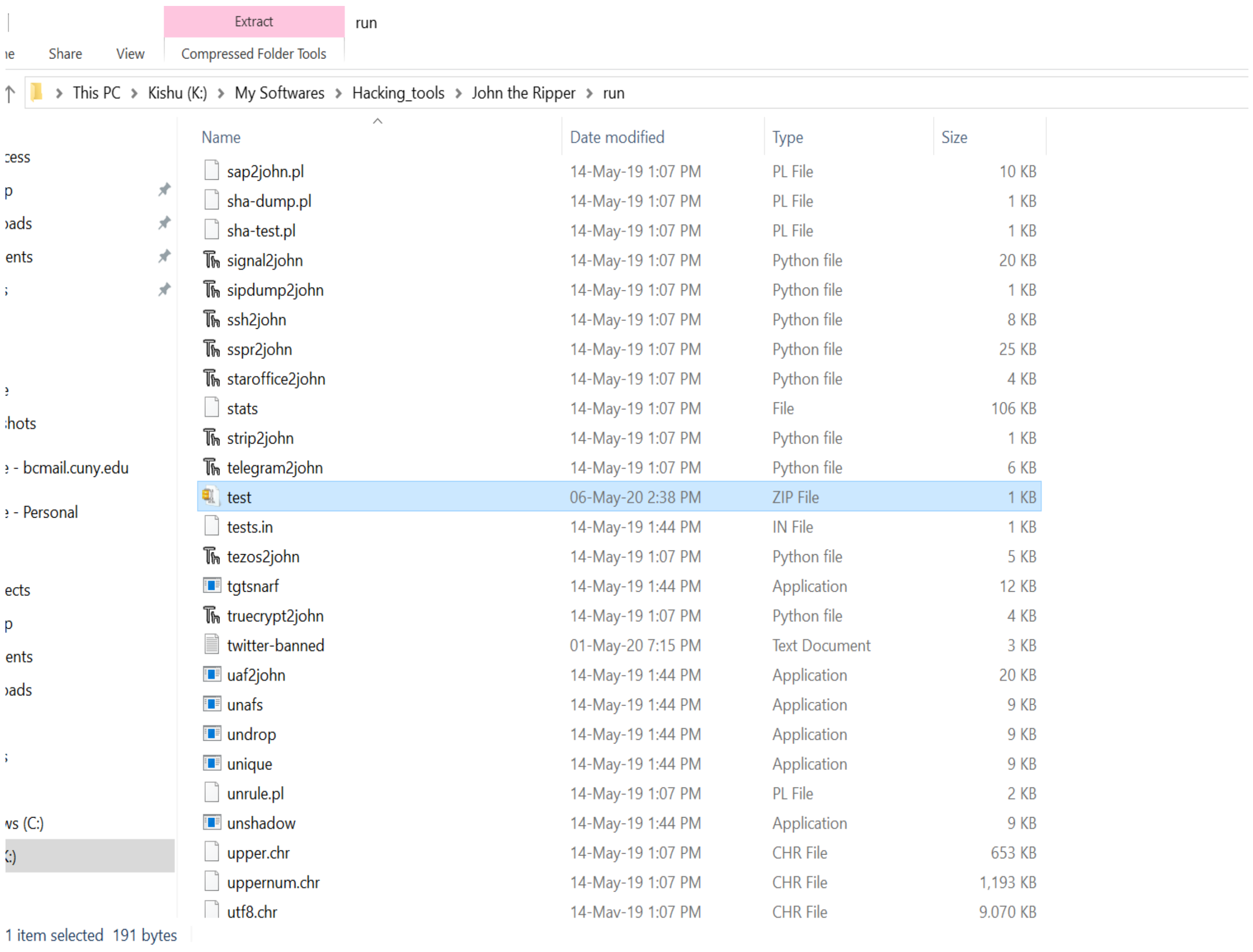
Enter password

Archiving with password

Enter password: Kishu

- ✓ Show password
- ☐ Encrypt file names

(3) Now Here I copy test.zip file in the run folder of the JtR software. So that we don't have to give test.zip file's location every time in our commands.

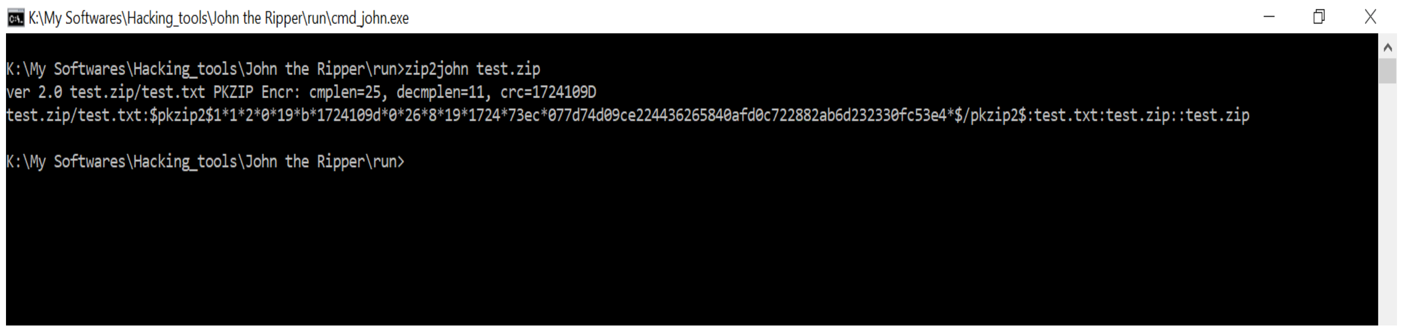


Extract
run
Share
View
Compressed Folder Tools
This PC > Kishu (K:) > My Softwares > Hacking_tools > John the Ripper > run
1 item selected 191 bytes

Name	Date modified	Type	Size
sap2john.pl	14-May-19 1:07 PM	PL File	10 KB
sha-dump.pl	14-May-19 1:07 PM	PL File	1 KB
sha-test.pl	14-May-19 1:07 PM	PL File	1 KB
signal2john	14-May-19 1:07 PM	Python file	20 KB
sipdump2john	14-May-19 1:07 PM	Python file	1 KB
ssh2john	14-May-19 1:07 PM	Python file	8 KB
sspr2john	14-May-19 1:07 PM	Python file	25 KB
staroffice2john	14-May-19 1:07 PM	Python file	4 KB
stats	14-May-19 1:07 PM	File	106 KB
strip2john	14-May-19 1:07 PM	Python file	1 KB
telegram2john	14-May-19 1:07 PM	Python file	6 KB
test	06-May-20 2:38 PM	ZIP File	1 KB
tests.in	14-May-19 1:44 PM	IN File	1 KB
tezos2john	14-May-19 1:07 PM	Python file	5 KB
tgtsnarf	14-May-19 1:44 PM	Application	12 KB
truecrypt2john	14-May-19 1:07 PM	Python file	4 KB
twitter-banned	01-May-20 7:15 PM	Text Document	3 KB
uaf2john	14-May-19 1:44 PM	Application	20 KB
unafs	14-May-19 1:44 PM	Application	9 KB
undrop	14-May-19 1:44 PM	Application	9 KB
unique	14-May-19 1:44 PM	Application	9 KB
unrule.pl	14-May-19 1:07 PM	PL File	2 KB
unshadow	14-May-19 1:44 PM	Application	9 KB
upper.chr	14-May-19 1:07 PM	CHR File	653 KB
uppernum.chr	14-May-19 1:07 PM	CHR File	1,193 KB
utf8.chr	14-May-19 1:07 PM	CHR File	9,070 KB

(4) Now I run this command to check whether we are getting a hash value of the password of test.zip file or not.

➢ Command : zip2john test.zip

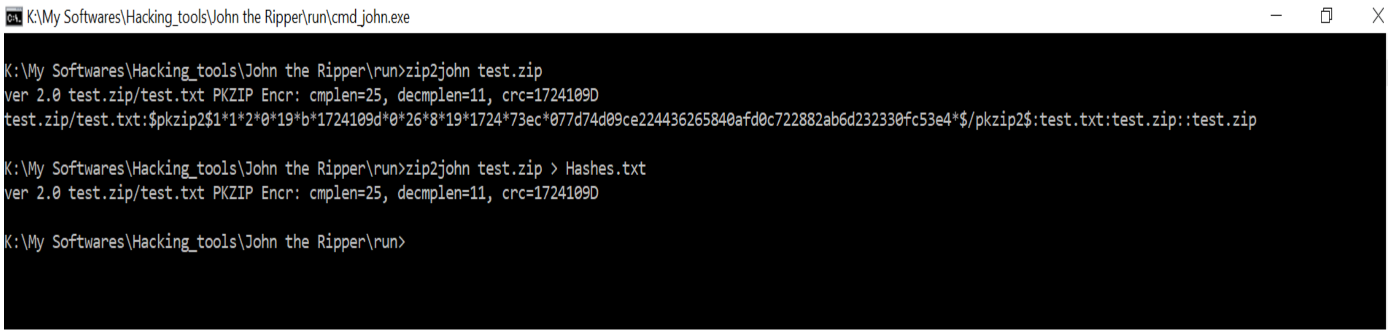


K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*077d74d09ce224436265840afd0c722882ab6d232330fc53e4*$/pkzip2$:test.txt:test.zip::test.zip

K:\My Softwares\Hacking_tools\John the Ripper\run>
```

(5) Now I store this hash value in a Hashes.txt file using this command.

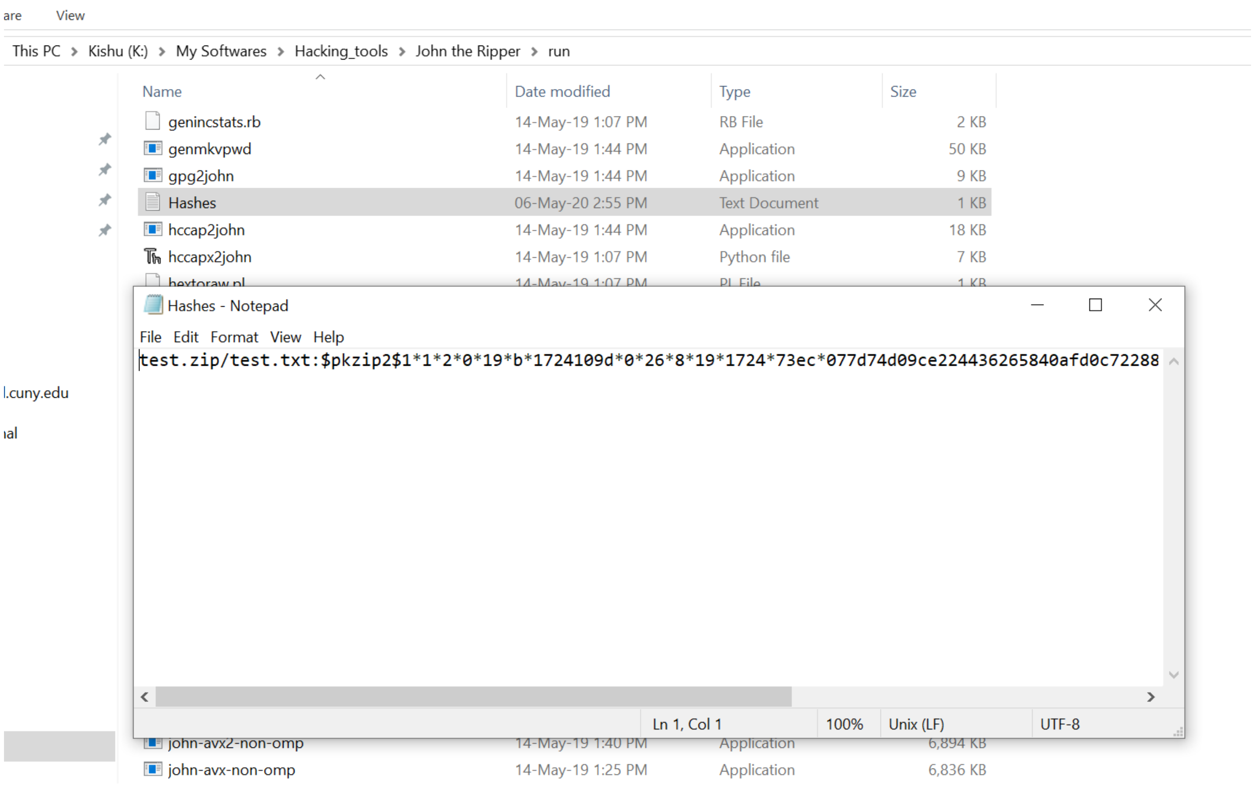
- ➢ Command : zip2john test.zip > Hashes.txt



K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*077d74d09ce224436265840afd0c722882ab6d232330fc53e4*$/pkzip2$:test.txt:test.zip::test.zip

K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip > Hashes.txt
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D

K:\My Softwares\Hacking_tools\John the Ripper\run>
```

(6) Hashes.txt file is generated now.


are
View
This PC > Kishu (K:) > My Softwares > Hacking_tools > John the Ripper > run
l.cuny.edu
ıal

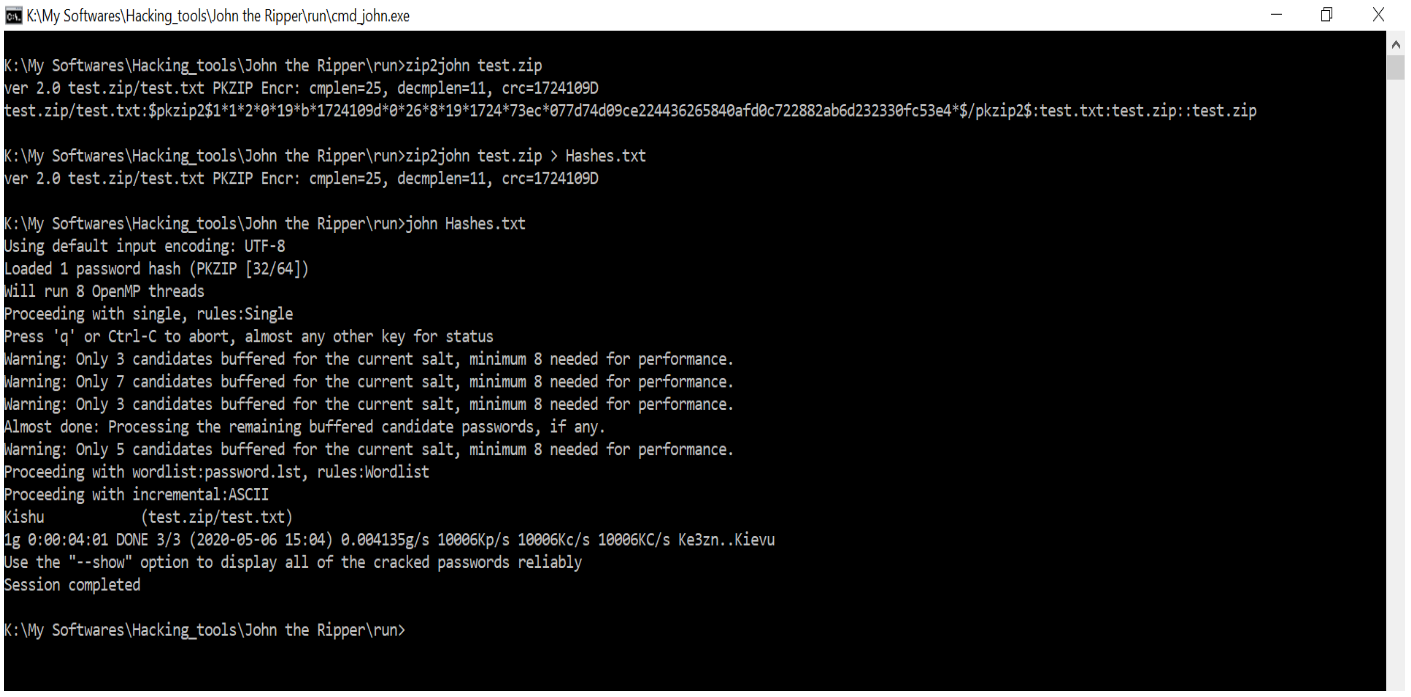
Name	Date modified	Type	Size
genincstats.rb	14-May-19 1:07 PM	RB File	2 KB
genmkvpwd	14-May-19 1:44 PM	Application	50 KB
gpg2john	14-May-19 1:44 PM	Application	9 KB
Hashes	06-May-20 2:55 PM	Text Document	1 KB
hccap2john	14-May-19 1:44 PM	Application	18 KB
hccapx2john	14-May-19 1:07 PM	Python file	7 KB
hextoraw.pl	14-May-19 1:07 PM	PL File	1 KB
john-avx2-non-omp	14-May-19 1:40 PM	Application	6,894 KB
john-avx-non-omp	14-May-19 1:25 PM	Application	6,836 KB

Hashes - Notepad

```
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*077d74d09ce224436265840afd0c72288
```

(7) Now I run this command to get the password of that file.

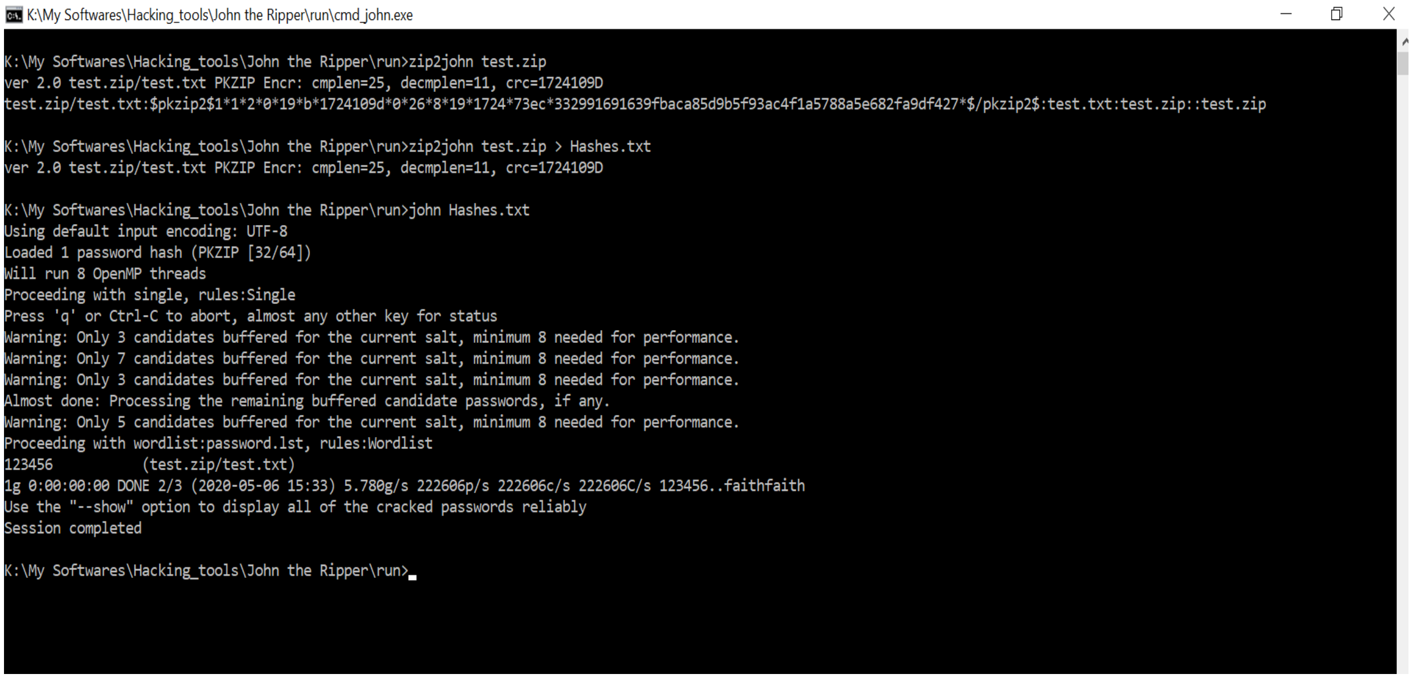
➢ Command : john Hashes.txt


K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*077d74d09ce224436265840afd0c722882ab6d232330fc53e4*$/pkzip2$:test.txt:test.zip::test.zip

K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip > Hashes.txt
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D

K:\My Softwares\Hacking_tools\John the Ripper\run>john Hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
Proceeding with incremental:ASCII
Kishu            (test.zip/test.txt)
1g 0:00:04:01 DONE 3/3 (2020-05-06 15:04) 0.004135g/s 10006Kp/s 10006Kc/s 10006KC/s Ke3zn..Kievu
Use the "--show" option to display all of the cracked passwords reliably
Session completed

K:\My Softwares\Hacking_tools\John the Ripper\run>
```

Here, we can see that first JtR proceeds with the single mode, then wordlist mode and at the end it proceeds with incremental mode because our password (Kishu) is somehow little bit difficult to crack. The time taken by the algorithm is 4 minutes. If I set any easy password, then it might only check with single mode and give us the output very quickly.

Let me take password = "123456" and now I am repeating all the steps for this as well.

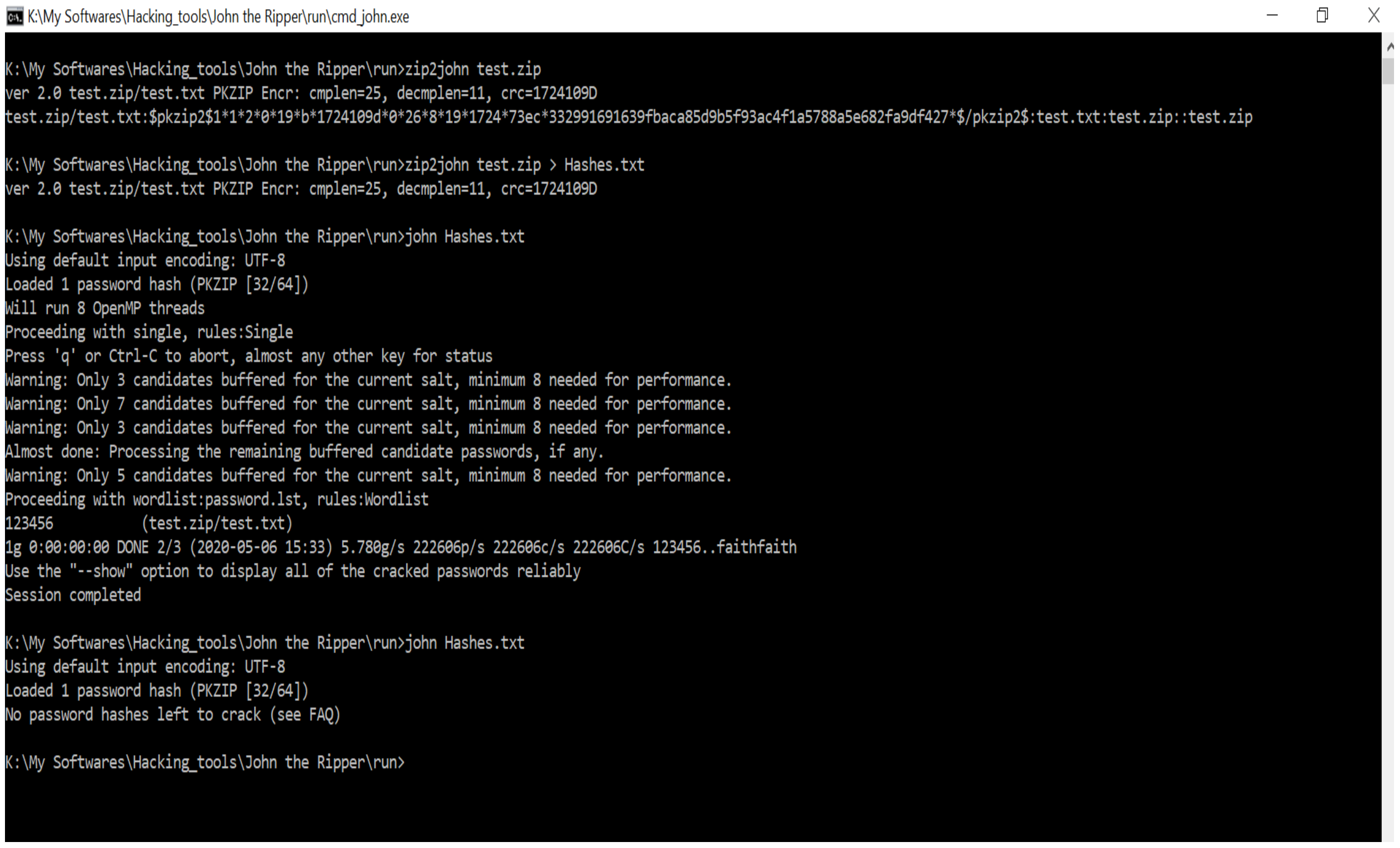

K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*332991691639fbaca85d9b5f93ac4f1a5788a5e682fa9df427*$/pkzip2$:test.txt:test.zip::test.zip

K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip > Hashes.txt
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D

K:\My Softwares\Hacking_tools\John the Ripper\run>john Hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
123456           (test.zip/test.txt)
1g 0:00:00:00 DONE 2/3 (2020-05-06 15:33) 5.780g/s 222606p/s 222606c/s 222606C/s 123456..faithfaith
Use the "--show" option to display all of the cracked passwords reliably
Session completed

K:\My Softwares\Hacking_tools\John the Ripper\run>
```

Here, we can see that JtR just took 2 modes to crack the password. In the wordlist mode, software has its own wordlist (password.lst) and it had a match for our password in it. So we get our password (123456) within no time because it is easy to crack.

Now, if I run this command again after getting the output password, it will give us below message.

- Command : john Hashes.txt



K:\My Softwares\Hacking_tools\John the Ripper\run\cmd_john.exe

```
K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D
test.zip/test.txt:$pkzip2$1*1*2*0*19*b*1724109d*0*26*8*19*1724*73ec*332991691639fbaca85d9b5f93ac4f1a5788a5e682fa9df427*$/pkzip2$:test.txt:test.zip::test.zip

K:\My Softwares\Hacking_tools\John the Ripper\run>zip2john test.zip > Hashes.txt
ver 2.0 test.zip/test.txt PKZIP Encr: cmplen=25, decmplen=11, crc=1724109D

K:\My Softwares\Hacking_tools\John the Ripper\run>john Hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
123456           (test.zip/test.txt)
1g 0:00:00:00 DONE 2/3 (2020-05-06 15:33) 5.780g/s 222606p/s 222606c/s 222606C/s 123456..faithfaith
Use the "--show" option to display all of the cracked passwords reliably
Session completed

K:\My Softwares\Hacking_tools\John the Ripper\run>john Hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
No password hashes left to crack (see FAQ)

K:\My Softwares\Hacking_tools\John the Ripper\run>
```

This is how we can crack different types of passwords using John the Ripper software. The main thing here is that we have the file (packet) to crack the password or to get the secure data. But what happens when we don’t have any package to crack? So, for getting packets from the network, I am using Wireshark software.

Wireshark

Wireshark is an application to capture and investigate network traffic. It is used for many reasons by many organizations. Here in this project, I will cover some basic things about the Wireshark and show a demonstration of how someone can get packet from the network and eventually get the Id and Password.

So, first of all, Wireshark is an open-source network protocol analysis software program started by Gerald Combs in 1998. Wireshark is absolutely safe to use. Government agencies, corporations, non-profits, and educational institutions use Wireshark for troubleshooting and teaching purposes.

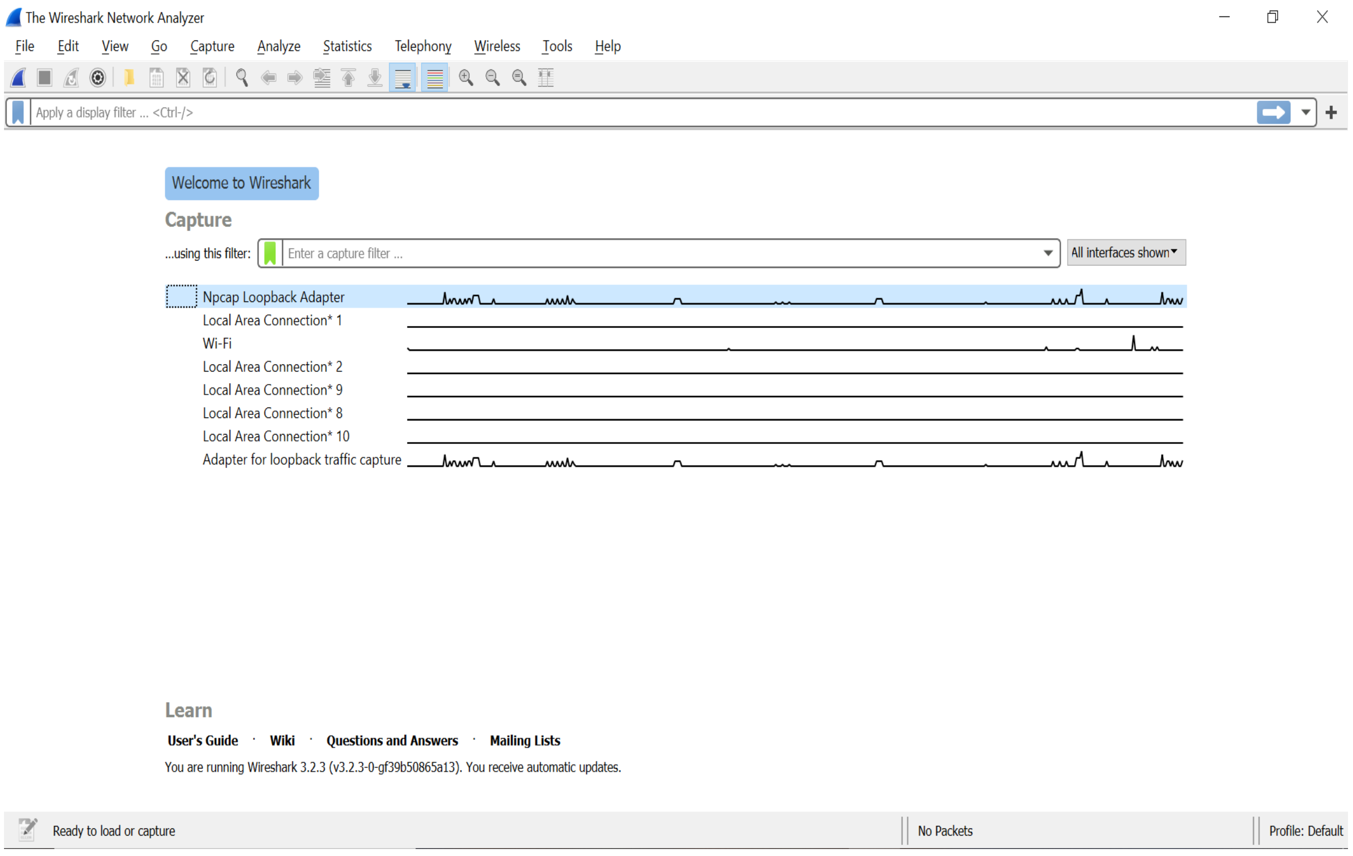
There are questions about the legality of Wireshark since it is a powerful packet sniffer. The Light side of the Force says that we should only use Wireshark on networks where we have permission to inspect network packets. Using Wireshark to look at packets without permission is a path to the Dark Side.

How does Wireshark work?

Wireshark captures network traffic on the local network and stores that data for offline analysis. It captures network traffic from Ethernet, Bluetooth, Wireless (IEEE.802.11), Token Ring, Frame Relay connections, and more.

Wireshark allows us to filter the log either before the capture starts or during analysis, so we can narrow down and zero into what you are looking for in the network trace. For example, we can set a filter to see TCP traffic between two IP addresses. We can set it only to show you the packets sent from one computer. The filters in Wireshark are one of the primary reasons it became the standard tool for packet analysis.

This is the homepage of the Wireshark.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

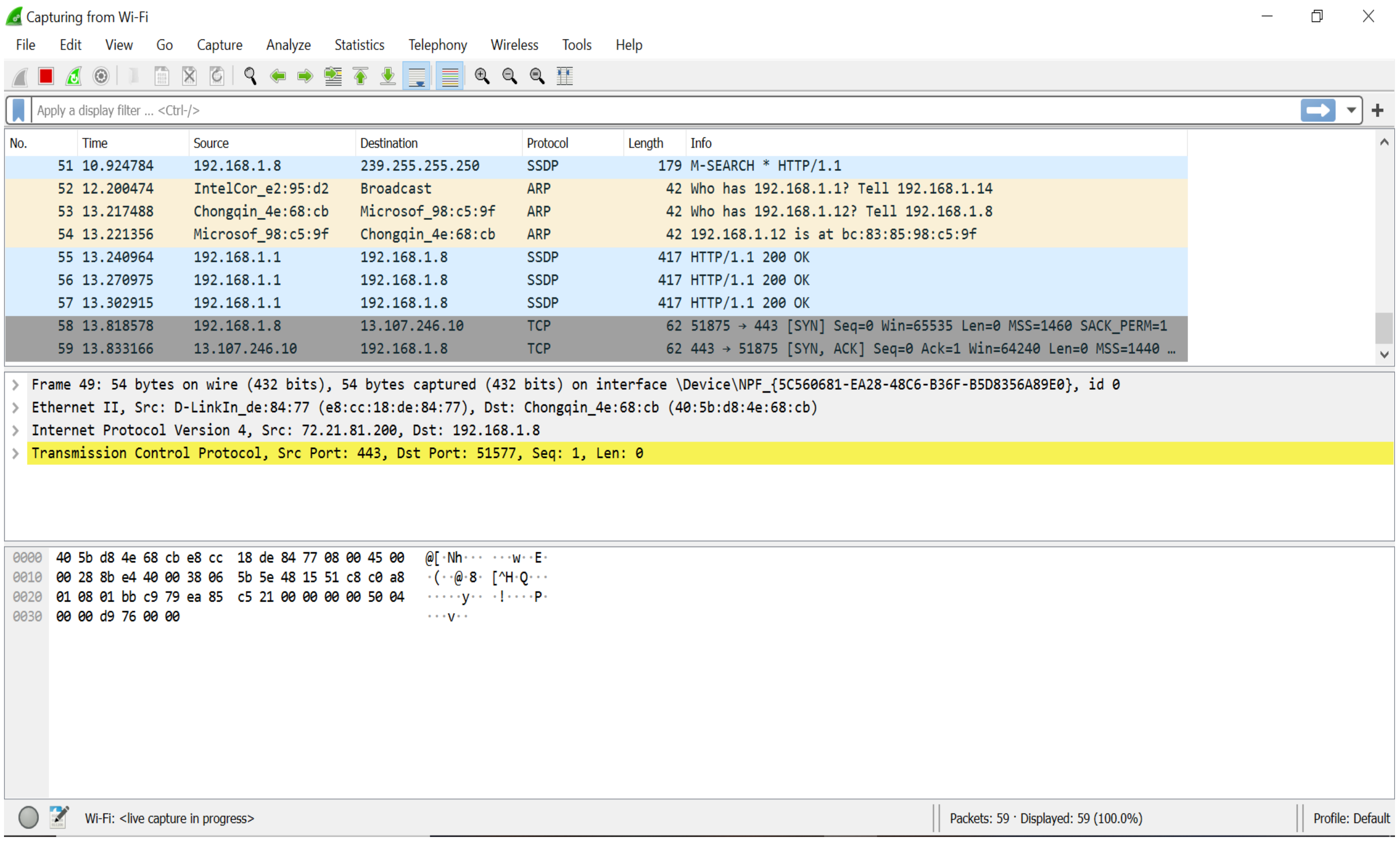
Npcap Loopback Adapter
Local Area Connection* 1
Wi-Fi
Local Area Connection* 2
Local Area Connection* 9
Local Area Connection* 8
Local Area Connection* 10
Adapter for loopback traffic capture

You are running Wireshark 3.2.3 (v3.2.3-0-gf39b50865a13). You receive automatic updates.

Ready to load or capture | No Packets | Profile: Default

When we open Wireshark, we see a screen that shows us a list of all of the network connections we can monitor. We also have a capture filter field, so we only capture the network traffic we want to see.

Here, I am getting all packets which are using Wi-Fi network.

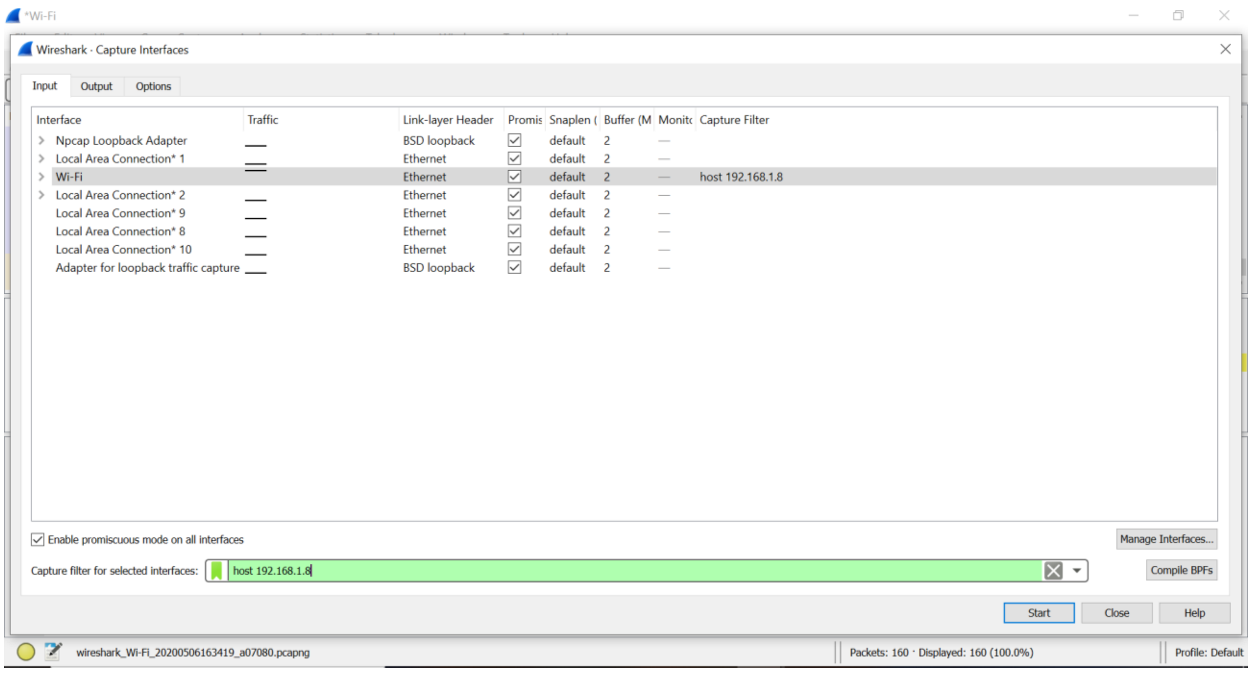

Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source              Destination         Protocol  Length Info
  51 10.924784  192.168.1.8         239.255.255.250     SSDP         179 M-SEARCH * HTTP/1.1
  52 12.200474  IntelCor_e2:95:d2   Broadcast           ARP           42 Who has 192.168.1.1? Tell 192.168.1.14
  53 13.217488  Chongqin_4e:68:cb   Microsof_98:c5:9f   ARP           42 Who has 192.168.1.12? Tell 192.168.1.8
  54 13.221356  Microsof_98:c5:9f   Chongqin_4e:68:cb   ARP           42 192.168.1.12 is at bc:83:85:98:c5:9f
  55 13.240964  192.168.1.1         192.168.1.8         SSDP         417 HTTP/1.1 200 OK
  56 13.270975  192.168.1.1         192.168.1.8         SSDP         417 HTTP/1.1 200 OK
  57 13.302915  192.168.1.1         192.168.1.8         SSDP         417 HTTP/1.1 200 OK
  58 13.818578  192.168.1.8         13.107.246.10       TCP           62 51875 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
  59 13.833166  13.107.246.10       192.168.1.8         TCP           62 443 → 51875 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 …

> Frame 49: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{5C560681-EA28-48C6-B36F-B5D8356A89E0}, id 0
> Ethernet II, Src: D-LinkIn_de:84:77 (e8:cc:18:de:84:77), Dst: Chongqin_4e:68:cb (40:5b:d8:4e:68:cb)
> Internet Protocol Version 4, Src: 72.21.81.200, Dst: 192.168.1.8
> Transmission Control Protocol, Src Port: 443, Dst Port: 51577, Seq: 1, Len: 0

0000  40 5b d8 4e 68 cb e8 cc  18 de 84 77 08 00 45 00   @[·Nh···  ···w··E·
0010  00 28 8b e4 40 00 38 06  5b 5e 48 15 51 c8 c0 a8   ·(··@·8·  [^H·Q···
0020  01 08 01 bb c9 79 ea 85  c5 21 00 00 00 00 50 04   ·····y··  ·!····P·
0030  00 00 d9 76 00 00                                  ···v··

Wi-Fi: <live capture in progress>    Packets: 59 · Displayed: 59 (100.0%)    Profile: Default
```

We can select one or more of the network interfaces using “shift left-click.” Once we have the network interface selected, we can start capturing packets.

So, for the example, I will capture a http package of a site and then I will analyze it to get some private data from it.

(1) First, I filter the search only for my laptop because right now many devices are connected in my wifi.


*Wi-Fi

Wireshark · Capture Interfaces

Input | Output | Options

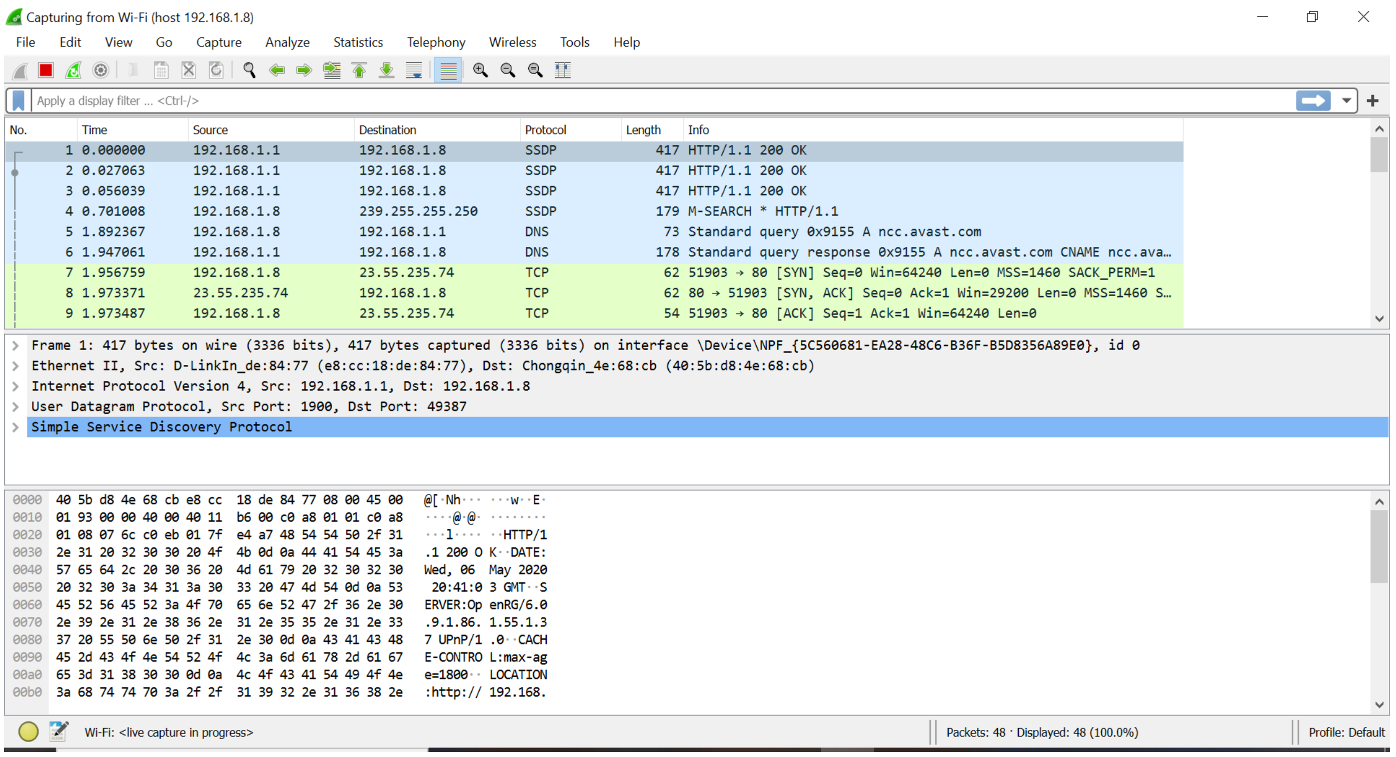
Interface	Traffic	Link-layer Header	Promis	Snaplen (	Buffer (M	Monitc	Capture Filter
Npcap Loopback Adapter		BSD loopback	✓	default	2	—	
Local Area Connection* 1		Ethernet	✓	default	2	—	
Wi-Fi		Ethernet	✓	default	2	—	host 192.168.1.8
Local Area Connection* 2		Ethernet	✓	default	2	—	
Local Area Connection* 9		Ethernet	✓	default	2	—	
Local Area Connection* 8		Ethernet	✓	default	2	—	
Local Area Connection* 10		Ethernet	✓	default	2	—	
Adapter for loopback traffic capture		BSD loopback	✓	default	2	—	

✓ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: host 192.168.1.8

wireshark_Wi-Fi_20200506163419_a07080.pcapng | Packets: 160 · Displayed: 160 (100.0%) | Profile: Default

(2) This is the network traffic of my laptop where 192.168.1.8 is my laptop's IP and 192.168.1.1 is my router's IP.


Capturing from Wi-Fi (host 192.168.1.8)

Apply a display filter ... <Ctrl-/>

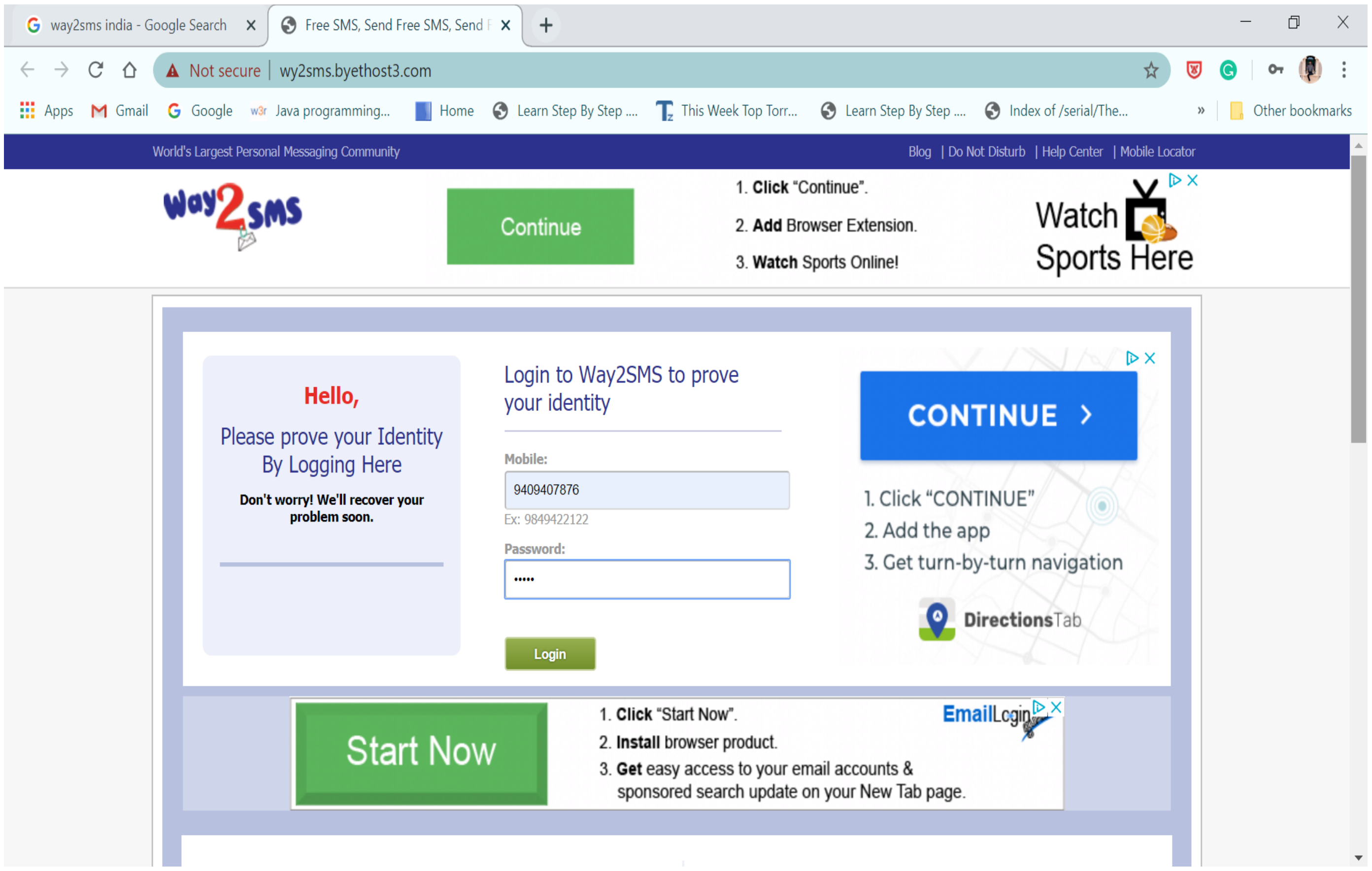
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.1	192.168.1.8	SSDP	417	HTTP/1.1 200 OK
2	0.027063	192.168.1.1	192.168.1.8	SSDP	417	HTTP/1.1 200 OK
3	0.056039	192.168.1.1	192.168.1.8	SSDP	417	HTTP/1.1 200 OK
4	0.701008	192.168.1.8	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
5	1.892367	192.168.1.8	192.168.1.1	DNS	73	Standard query 0x9155 A ncc.avast.com
6	1.947061	192.168.1.1	192.168.1.8	DNS	178	Standard query response 0x9155 A ncc.avast.com CNAME ncc.ava…
7	1.956759	192.168.1.8	23.55.235.74	TCP	62	51903 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
8	1.973371	23.55.235.74	192.168.1.8	TCP	62	80 → 51903 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 S…
9	1.973487	192.168.1.8	23.55.235.74	TCP	54	51903 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0

```
> Frame 1: 417 bytes on wire (3336 bits), 417 bytes captured (3336 bits) on interface \Device\NPF_{5C560681-EA28-48C6-B36F-B5D8356A89E0}, id 0
> Ethernet II, Src: D-LinkIn_de:84:77 (e8:cc:18:de:84:77), Dst: Chongqin_4e:68:cb (40:5b:d8:4e:68:cb)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.8
> User Datagram Protocol, Src Port: 1900, Dst Port: 49387
> Simple Service Discovery Protocol
```

```
0000  40 5b d8 4e 68 cb e8 cc  18 de 84 77 08 00 45 00   @[·Nh··· ···w··E·
0010  01 93 00 00 40 00 40 11  b6 00 c0 a8 01 01 c0 a8   ····@·@· ········
0020  01 08 07 6c c0 eb 01 7f  e4 a7 48 54 54 50 2f 31   ···l···· ··HTTP/1
0030  2e 31 20 32 30 30 20 4f  4b 0d 0a 44 41 54 45 3a   .1 200 O K··DATE:
0040  57 65 64 2c 20 30 36 20  4d 61 79 20 32 30 32 30   Wed, 06  May 2020
0050  20 32 30 3a 34 31 3a 30  33 20 47 4d 54 0d 0a 53    20:41:0 3 GMT··S
0060  45 52 56 45 52 3a 4f 70  65 6e 52 47 2f 36 2e 30   ERVER:Op enRG/6.0
0070  2e 39 2e 31 2e 38 36 2e  31 2e 35 35 2e 31 2e 33   .9.1.86. 1.55.1.3
0080  37 20 55 50 6e 50 2f 31  2e 30 0d 0a 43 41 43 48   7 UPnP/1 .0··CACH
0090  45 2d 43 4f 4e 54 52 4f  4c 3a 6d 61 78 2d 61 67   E-CONTRO L:max-ag
00a0  65 3d 31 38 30 30 0d 0a  4c 4f 43 41 54 49 4f 4e   e=1800·· LOCATION
00b0  3a 68 74 74 70 3a 2f 2f  31 39 32 2e 31 36 38 2e   :http:// 192.168.
```

Wi-Fi: <live capture in progress> | Packets: 48 · Displayed: 48 (100.0%) | Profile: Default

(3) Now, I will enter some credentials like id and password in a random site's login page. The main thing here is that the site is not secure. So that, the packet that we will take from the network will not be encrypted basically. Here, I entered Mobile number = "9409407876" and Password = "Kishu".



way2sms india - Google Search
Free SMS, Send Free SMS, Send F
Not secure | wy2sms.byethost3.com
Apps
Gmail
Google
Java programming...
Home
Learn Step By Step ....
This Week Top Torr...
Learn Step By Step ....
Index of /serial/The...
Other bookmarks
World's Largest Personal Messaging Community
Blog | Do Not Disturb | Help Center | Mobile Locator
Continue
1. Click "Continue".
2. Add Browser Extension.
3. Watch Sports Online!
Watch Sports Here
CONTINUE
1. Click "CONTINUE"
2. Add the app
3. Get turn-by-turn navigation
DirectionsTab
Start Now
1. Click "Start Now".
2. Install browser product.
3. Get easy access to your email accounts & sponsored search update on your New Tab page.
EmailLogin

Hello,

Please prove your Identity By Logging Here

Don't worry! We'll recover your problem soon.

Login to Way2SMS to prove your identity

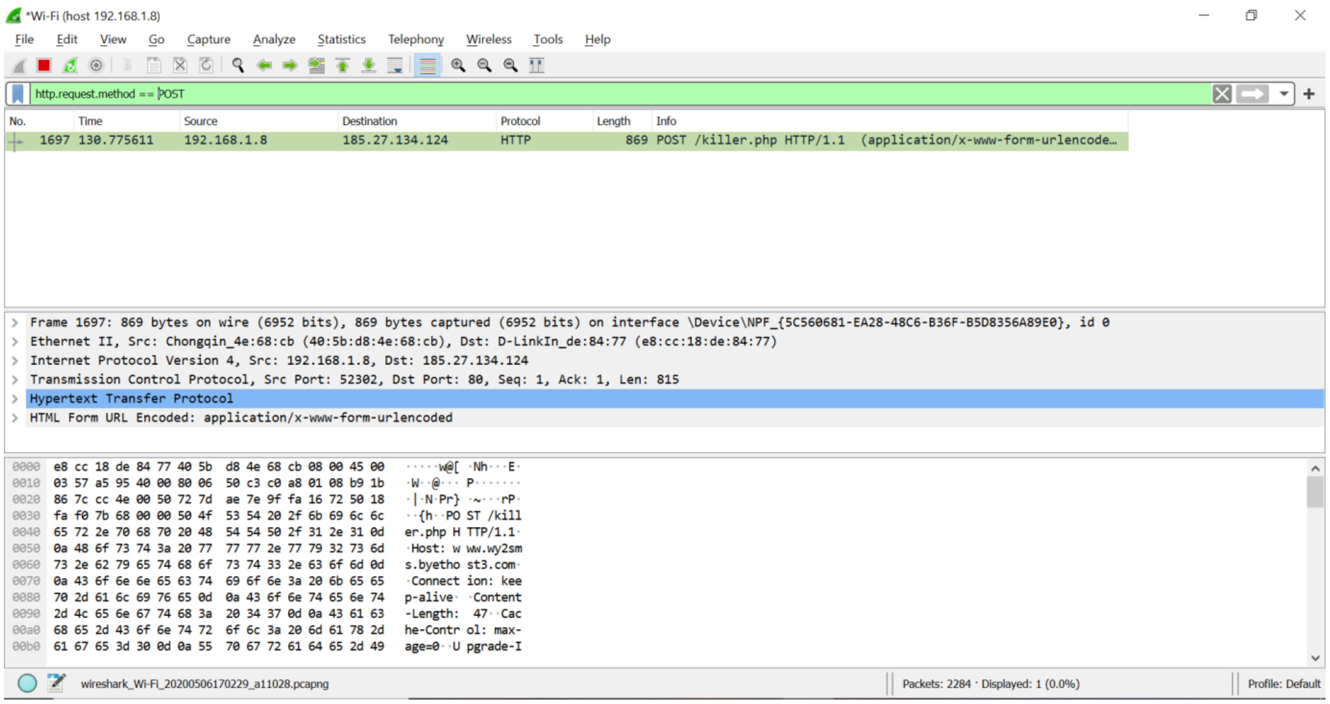
Mobile: 9409407876

Ex: 9849422122

Password: •••••

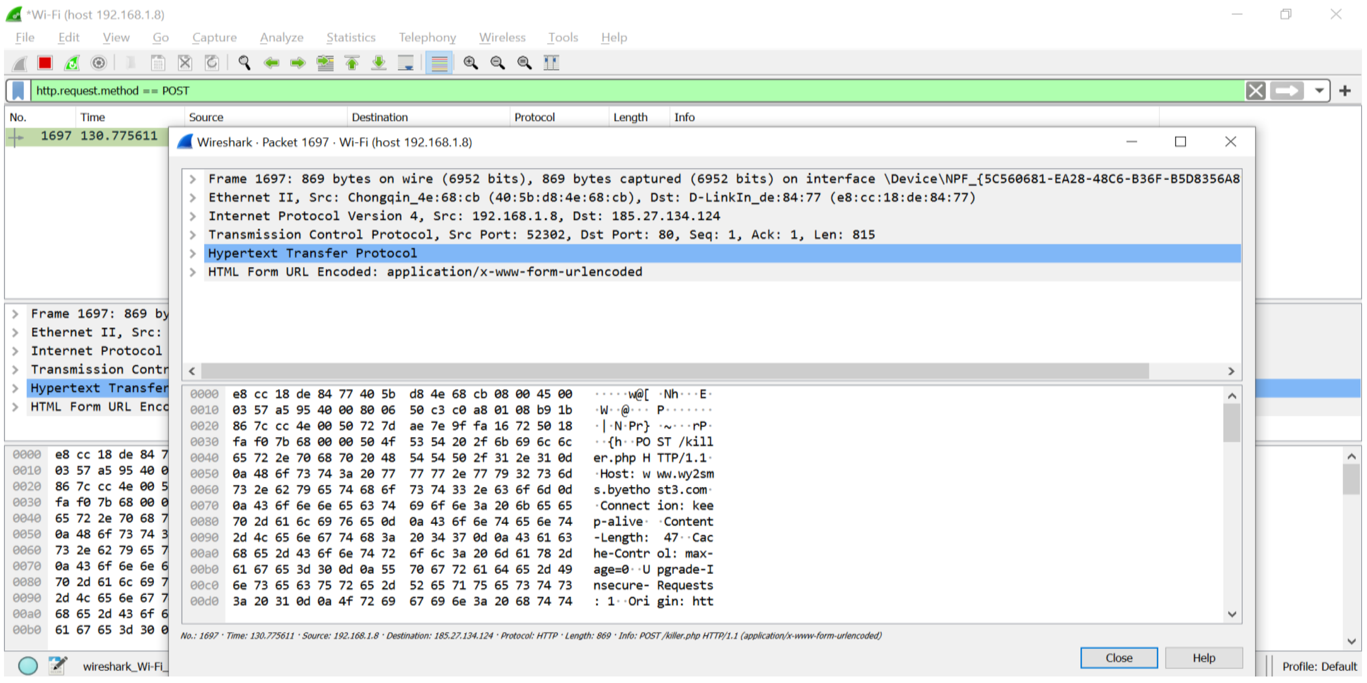
Login

(4) After entering Mobile number and Password in this site, when we click on Login button, http post method will come into the picture. Till now, we already got many network traffic and it keeps getting new packets from the network. So, we again filter our search and write “http.request.method == POST” in the display filter.


*Wi-Fi (host 192.168.1.8)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
No. Time Source Destination Protocol Length Info

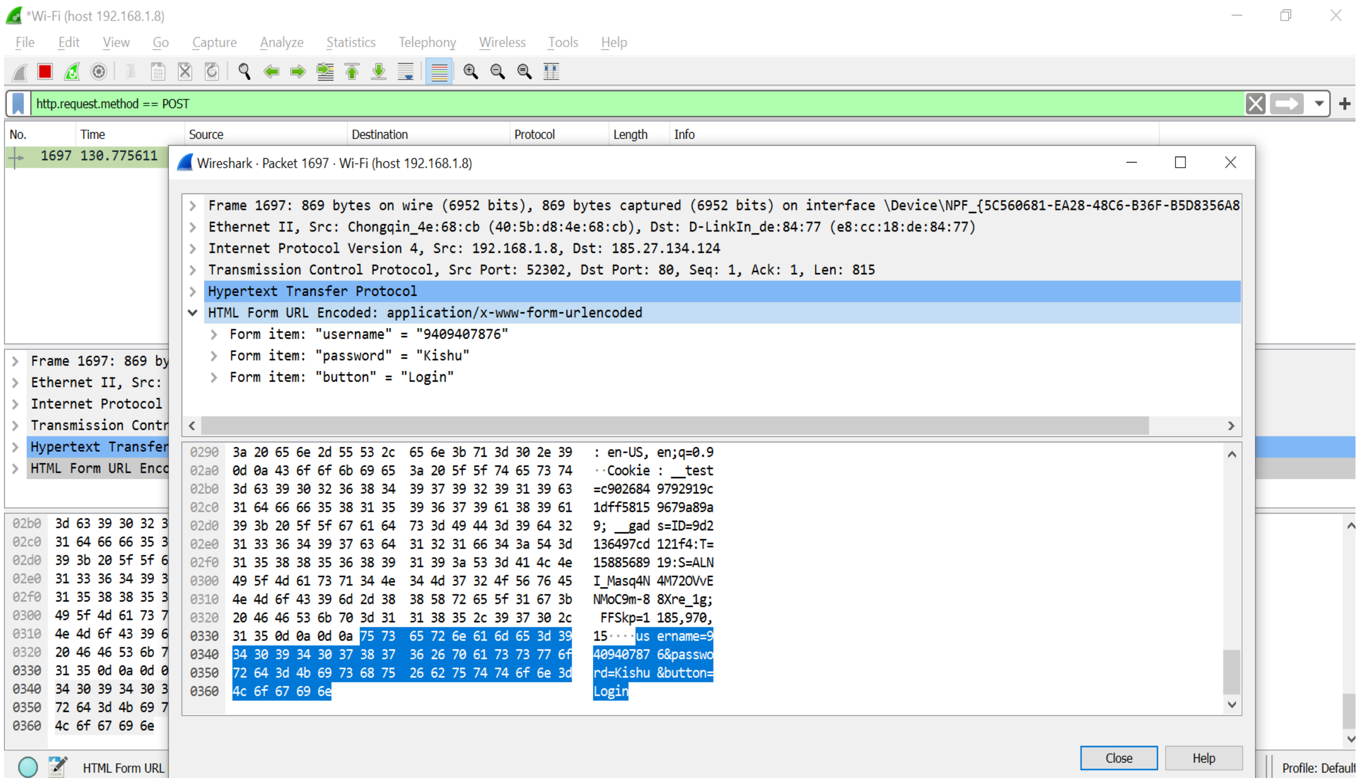
```
http.request.method == POST

1697 130.775611   192.168.1.8   185.27.134.124   HTTP   869 POST /killer.php HTTP/1.1  (application/x-www-form-urlencode…

> Frame 1697: 869 bytes on wire (6952 bits), 869 bytes captured (6952 bits) on interface \Device\NPF_{5C560681-EA28-48C6-B36F-B5D8356A89E0}, id 0
> Ethernet II, Src: Chongqin_4e:68:cb (40:5b:d8:4e:68:cb), Dst: D-LinkIn_de:84:77 (e8:cc:18:de:84:77)
> Internet Protocol Version 4, Src: 192.168.1.8, Dst: 185.27.134.124
> Transmission Control Protocol, Src Port: 52302, Dst Port: 80, Seq: 1, Ack: 1, Len: 815
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded

0000  e8 cc 18 de 84 77 40 5b  d8 4e 68 cb 08 00 45 00   ·····w@[ ·Nh···E·
0010  03 57 a5 95 40 00 80 06  50 c3 c0 a8 01 08 b9 1b   ·W··@··· P·······
0020  86 7c cc 4e 00 50 72 7d  ae 7e 9f fa 16 72 50 18   ·|·N·Pr} ·~···rP·
0030  fa f0 7b 68 00 00 50 4f  53 54 20 2f 6b 69 6c 6c   ··{h··PO ST /kill
0040  65 72 2e 70 68 70 20 48  54 54 50 2f 31 2e 31 0d   er.php H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 77  77 77 2e 77 79 32 73 6d   ·Host: w ww.wy2sm
0060  73 2e 62 79 65 74 68 6f  73 74 33 2e 63 6f 6d 0d   s.byetho st3.com·
0070  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 6b 65 65   ·Connect ion: kee
0080  70 2d 61 6c 69 76 65 0d  0a 43 6f 6e 74 65 6e 74   p-alive· ·Content
0090  2d 4c 65 6e 67 74 68 3a  20 34 37 0d 0a 43 61 63   -Length:  47··Cac
00a0  68 65 2d 43 6f 6e 74 72  6f 6c 3a 20 6d 61 78 2d   he-Contr ol: max-
00b0  61 67 65 3d 30 0d 0a 55  70 67 72 61 64 65 2d 49   age=0··U pgrade-I

wireshark_Wi-Fi_20200506170229_a11028.pcapng    Packets: 2284 · Displayed: 1 (0.0%)    Profile: Default
```

(5) Now, double click on the HTTP protocol packet which we just filtered. We will see a window, from which we can get many information regarding that packet.


*Wi-Fi (host 192.168.1.8)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
No. Time Source Destination Protocol Length Info
Wireshark · Packet 1697 · Wi-Fi (host 192.168.1.8)
Close
Help
Profile: Default

```
http.request.method == POST

1697 130.775611

> Frame 1697: 869 bytes on wire (6952 bits), 869 bytes captured (6952 bits) on interface \Device\NPF_{5C560681-EA28-48C6-B36F-B5D8356A8
> Ethernet II, Src: Chongqin_4e:68:cb (40:5b:d8:4e:68:cb), Dst: D-LinkIn_de:84:77 (e8:cc:18:de:84:77)
> Internet Protocol Version 4, Src: 192.168.1.8, Dst: 185.27.134.124
> Transmission Control Protocol, Src Port: 52302, Dst Port: 80, Seq: 1, Ack: 1, Len: 815
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded

0000  e8 cc 18 de 84 77 40 5b  d8 4e 68 cb 08 00 45 00   ·····w@[ ·Nh···E·
0010  03 57 a5 95 40 00 80 06  50 c3 c0 a8 01 08 b9 1b   ·W··@··· P·······
0020  86 7c cc 4e 00 50 72 7d  ae 7e 9f fa 16 72 50 18   ·|·N·Pr} ·~···rP·
0030  fa f0 7b 68 00 00 50 4f  53 54 20 2f 6b 69 6c 6c   ··{h··PO ST /kill
0040  65 72 2e 70 68 70 20 48  54 54 50 2f 31 2e 31 0d   er.php H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 77  77 77 2e 77 79 32 73 6d   ·Host: w ww.wy2sm
0060  73 2e 62 79 65 74 68 6f  73 74 33 2e 63 6f 6d 0d   s.byetho st3.com·
0070  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 6b 65 65   ·Connect ion: kee
0080  70 2d 61 6c 69 76 65 0d  0a 43 6f 6e 74 65 6e 74   p-alive· ·Content
0090  2d 4c 65 6e 67 74 68 3a  20 34 37 0d 0a 43 61 63   -Length:  47··Cac
00a0  68 65 2d 43 6f 6e 74 72  6f 6c 3a 20 6d 61 78 2d   he-Contr ol: max-
00b0  61 67 65 3d 30 0d 0a 55  70 67 72 61 64 65 2d 49   age=0··U pgrade-I
00c0  6e 73 65 63 75 72 65 2d  52 65 71 75 65 73 74 73   nsecure- Requests
00d0  3a 20 31 0d 0a 4f 72 69  67 69 6e 3a 20 68 74 74   : 1··Ori gin: htt

No.: 1697 · Time: 130.775611 · Source: 192.168.1.8 · Destination: 185.27.134.124 · Protocol: HTTP · Length: 869 · Info: POST /killer.php HTTP/1.1 (application/x-www-form-urlencoded)
```

(6) Now, it has a lot of information, but we are looking for the phone number and password. For that, go into HTML Form URL Encoded.



```
http.request.method == POST

No.    Time
1697   130.775611

Wireshark · Packet 1697 · Wi-Fi (host 192.168.1.8)

> Frame 1697: 869 bytes on wire (6952 bits), 869 bytes captured (6952 bits) on interface \Device\NPF_{5C560681-EA28-48C6-B36F-B5D8356A8
> Ethernet II, Src: Chongqin_4e:68:cb (40:5b:d8:4e:68:cb), Dst: D-LinkIn_de:84:77 (e8:cc:18:de:84:77)
> Internet Protocol Version 4, Src: 192.168.1.8, Dst: 185.27.134.124
> Transmission Control Protocol, Src Port: 52302, Dst Port: 80, Seq: 1, Ack: 1, Len: 815
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "9409407876"
  > Form item: "password" = "Kishu"
  > Form item: "button" = "Login"

0290  3a 20 65 6e 2d 55 53 2c  65 6e 3b 71 3d 30 2e 39   : en-US, en;q=0.9
02a0  0d 0a 43 6f 6f 6b 69 65  3a 20 5f 5f 74 65 73 74   ··Cookie : __test
02b0  3d 63 39 30 32 36 38 34  39 37 39 32 39 31 39 63   =c902684 9792919c
02c0  31 64 66 66 35 38 31 35  39 36 37 39 61 38 39 61   1dff5815 9679a89a
02d0  39 3b 20 5f 5f 67 61 64  73 3d 49 44 3d 39 64 32   9; __gad s=ID=9d2
02e0  31 33 36 34 39 37 63 64  31 32 31 66 34 3a 54 3d   136497cd 121f4:T=
02f0  31 35 38 38 35 36 38 39  31 39 3a 53 3d 41 4c 4e   15885689 19:S=ALN
0300  49 5f 4d 61 73 71 34 4e  34 4d 37 32 4f 56 76 45   I_Masq4N 4M72OVvE
0310  4e 4d 6f 43 39 6d 2d 38  38 58 72 65 5f 31 67 3b   NMoC9m-8 8Xre_1g;
0320  20 46 46 53 6b 70 3d 31  31 38 35 2c 39 37 30 2c    FFSkp=1 185,970,
0330  31 35 0d 0a 0d 0a 75 73  65 72 6e 61 6d 65 3d 39   15····us ername=9
0340  34 30 39 34 30 37 38 37  36 26 70 61 73 73 77 6f   40940787 6&passwo
0350  72 64 3d 4b 69 73 68 75  26 62 75 74 74 6f 6e 3d   rd=Kishu &button=
0360  4c 6f 67 69 6e                                     Login
```

Here, we got the private data username = “9409407876” and password = “Kishu”. If the site is private and using low level of encryption, we can get the packets from that site and then decrypt them using John the Ripper.

So, my main goal for this project is to gain secure private data and understand how whole network works. Using Wireshark, we can get packets from the network which we are connected. After getting packages from the network, we can just decrypt them or crack their passwords using John the Ripper.

I learned many things from these 2 hacking tools, like how I check how many devices are connected in my wifi, what people are surfing on internet in the same wifi, how routers connect our devices to the internet, what happens when we search any site (facebook.com), how we can analyze the packet, how we can crack zip or rar file’s password, how we can get password from different types of hashes, and many more. Fundamentally, if we combine John the Ripper and Wireshark, we can hack many stuffs.