

# Wireshark Basics

414C504F

- 
- Traffic capture and traffic filtering with Wireshark
  - SSL ManInTheMiddle with Wireshark
  - WLAN traffic ManInTheMiddle with Wireshark

- 
- Packet analyser / traffic sniffer
  - Open-source
  - Cross-platform
  - Fancy GUI
  - <https://www.wireshark.org/>

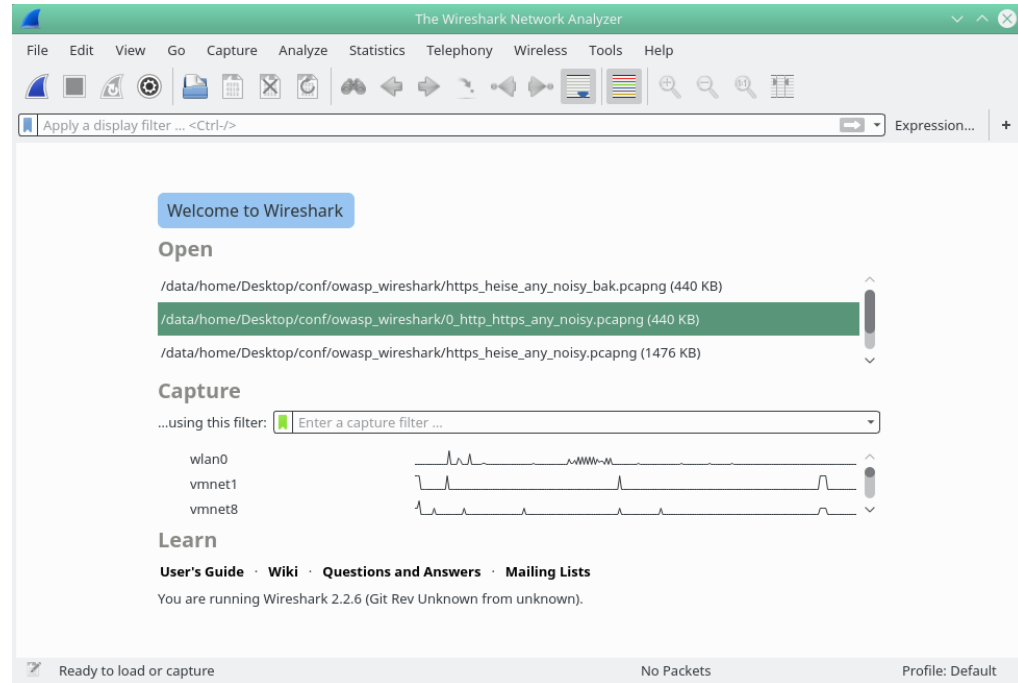
# Data packets capturing

To start capturing

- Select a network interface
- Click on the blue shark fin button / press Ctrl + E

To stop capturing

- Click on the red stop button / press Ctrl + E



# Data packets capturing

No.	Time	Source	Destination	Protocol	Lengt	Info
13	1.256128834	192.168.178.27	193.70.91.56	HTTP	183	GET / HTTP/1.1
14	1.272047640	193.70.91.56	192.168.178.27	TCP	68	80 → 55590 [ACK] Seq=1 Ack=116 Win=14592 Le
15	1.503106801	192.168.178.27	88.98.79.97	UDPENC...	45	NAT-keepalive
16	2.992854539	192.168.178.27	88.98.79.97	ESP	144	ESP (SPI=0xfc38d29f)
17	3.732746500	193.70.91.56	192.168.178.27	HTTP	345	HTTP/1.1 301 Moved Permanently
18	3.732785793	193.70.91.56	192.168.178.27	TCP	68	80 → 55590 [FIN, ACK] Seq=278 Ack=116 Win=1
19	3.732809890	192.168.178.27	193.70.91.56	TCP	68	55590 → 80 [ACK] Seq=116 Ack=278 Win=30336

> Frame 10: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0

> Linux cooked capture

> Internet Protocol Version 4, Src: 192.168.178.27, Dst: 193.70.91.56

> Transmission Control Protocol, Src Port: 55590, Dst Port: 80, Seq: 0, Len: 0

```
0000  00 04 00 01 00 06 b8 08  cf 58 61 64 00 00 08 00  ..... .Xad....
0010  45 00 00 3c c9 c5 40 00  40 06 e1 b3 c0 a8 b2 1b  E..<..@. @.....
0020  c1 46 5b 38 d9 26 00 50  e6 3e 0e f3 00 00 00 00  .F[8.&.P .>.....
0030  a0 02 72 10 41 2c 00 00  02 04 05 b4 04 02 08 0a  ..r.A,.. .....
0040  61 f2 d4 e5 00 00 00 00  01 03 03 07  a..... .....
```

# Data packets capturing

Top frame:

Number | Time | Source | Destination | Protocol | Length | Info

# Data packets capturing

Top frame:

Number | Time | Source | Destination | Protocol | Length | Info

Middle frame example:

- > Frame
- > Linux cooked capture
- > Internet protocol version, source, destination
- > Transmission control protocol, src port, dst port, seq, len

# Data packets capturing

Top frame:

Number | Time | Source | Destination | Protocol | Length | Info

Middle frame example:

- > Frame
- > Linux cooked capture
- > Internet protocol version, source, destination
- > Transmission control protocol, src port, dst port, seq, len

Bottom frame:

Data



No.	Time	Source	Destination	Protocol	Length	Info
13	1.256128834	192.168.178.27	193.70.91.56	HTTP	183	GET / HTTP/1.1
14	1.272047640	193.70.91.56	192.168.178.27	TCP	68	80 → 55590 [ACK]
15	1.503106801	192.168.178.27	88.98.79.97	UDPENC...	45	NAT-keepalive
16	2.992854539	192.168.178.27	88.98.79.97	ESP	144	ESP (SPI=0xfc38d
17	2.722746500	192.70.01.56	192.168.178.27	HTTP	245	HTTP/1.1 201 Mov

> Transmission Control Protocol, Src Port: 55590, Dst Port: 80, Seq: 1, Ack: 1, Len: 115

▼ Hypertext Transfer Protocol

▼ GET / HTTP/1.1\r\n

> [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]

Request Method: GET

Request URI: /

```

0000 00 04 00 01 00 06 b8 08 cf 58 61 64 00 00 08 00 ..... .Xad....
0010 45 00 00 a7 c9 c7 40 00 40 06 e1 46 c0 a8 b2 1b E.....@. @..F....
0020 c1 46 5b 38 d9 26 00 50 e6 3e 0e f4 76 3e ff 39 .F[8.&.P .>..v>.9
0030 80 18 00 e5 1c ec 00 00 01 01 08 0a 61 f2 d4 e9 ..... ..a...
0040 df f7 4b 0b 47 45 54 20 2f 20 48 54 54 50 2f 31 ..K.GET / HTTP/1
0050 2e 31 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 .1..Acce pt-Encod
0060 69 6e 67 3a 20 69 64 65 6e 74 69 74 79 0d 0a 48 ing: ide ntity..H
0070 6f 73 74 3a 20 31 39 33 2e 37 30 2e 39 31 2e 35 ost: 193 .70.91.5
0080 36 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 63 6..Conne ction: c
0090 6c 6f 73 65 0d 0a 55 73 65 72 2d 41 67 65 6e 74 lose..Us er-Agent
00a0 3a 20 50 79 74 68 6f 6e 2d 75 72 6c 6c 69 62 2f : Python -urllib/
00b0 33 2e 35 0d 0a 0d 0a 3.5....

```

---

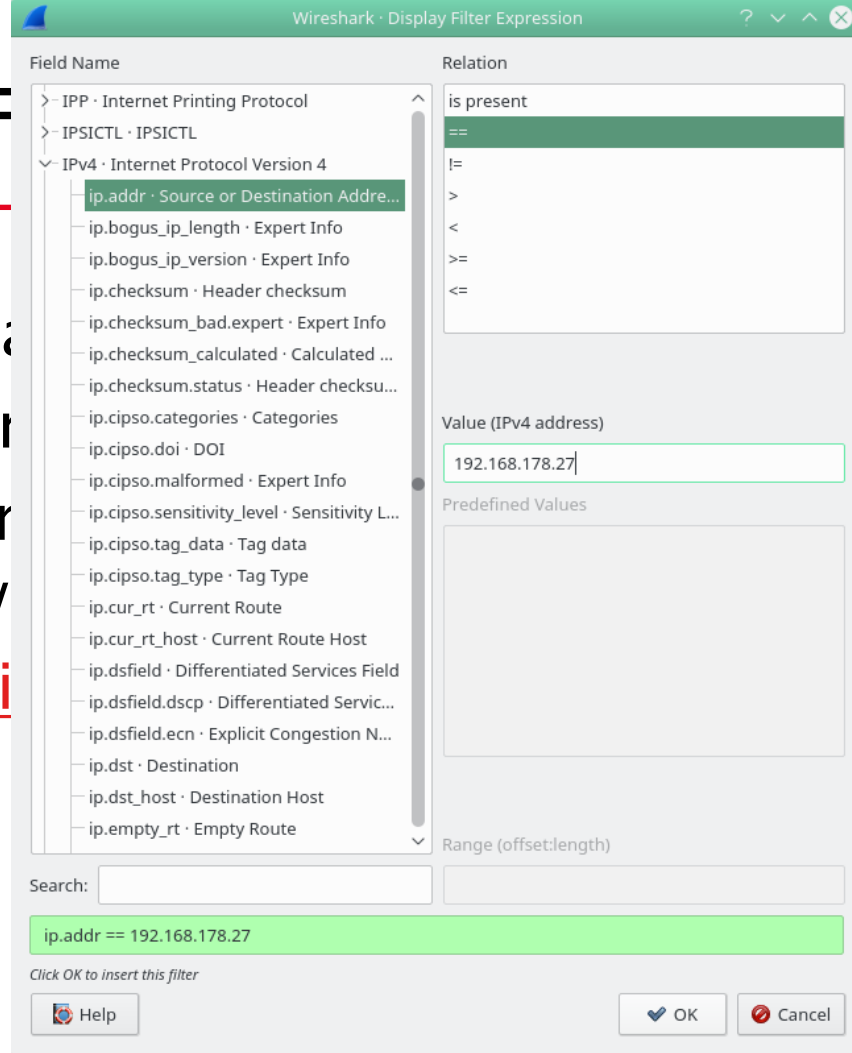
There are 2 ways to filter:

- Build a filter via the fancy GUI (Expression button)
- Type a filter into the “Apply a display filter” entry field (below the toolbar)
- <https://wiki.wireshark.org/DisplayFilters>

# Wireshark Filter

There are 2 ways

- Build a filter
- Type a filter
- [https://wiki](https://wiki.wireshark.org/DisplayFilterExpression)



context

on button)  
Filter" entry

S

# Wireshark Filters Relations

English	C-like	Description and example
eq	==	Equal. <code>ip.src==10.0.0.5</code>
ne	!=	Not equal. <code>ip.src!=10.0.0.5</code>
gt	>	Greater than. <code>frame.len &gt; 10</code>
lt	<	Less than. <code>frame.len &lt; 128</code>
ge	>=	Greater than or equal to. <code>frame.len ge 0x100</code>
le	<=	Less than or equal to. <code>frame.len le 0x20</code>
contains		Protocol, field or slice contains a value. <code>sip.To contains "a1762"</code>
matches	~	Protocol or text field match Perl regular expression. <code>http.host matches "acme\.(org com net)"</code>
bitwise_and	&	Compare bit field value. <code>tcp.flags &amp; 0x02</code>

# Wireshark Combining Expressions



English	C-like	Description and example
and	&&	Logical AND. <code>ip.src==10.0.0.5 and tcp.flags.fin</code>
or		Logical OR. <code>ip.scr==10.0.0.5 or ip.src==192.1.1.1</code>
xor	^^	Logical XOR. <code>tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29</code>
not	!	Logical NOT. <code>not llc</code>
[...]		Slice Operator. <code>eth.addr[0:3]==00:06:5B</code>
in		Membership Operator. <code>tcp.port in {80 443 8080}</code>

# Most common Wireshark filters

---

```
tcp.port eq 80  
tcp.srcport==443
```

Filter for HTTP and HTTPS traffic:

```
tcp.port==443 or tcp.port==80  
ssl or http  
tcp.port in {80 443 8080}  
tcp.port == 80 || tcp.port == 443 || tcp.port == 8080
```

# Most common Wireshark filters

---

Filter for a protocol:

tcp

udp

dns

IP addresses:

ip.addr == 10.43.54.65

! ( ip.addr == 10.43.54.65 )

Examples for web traffic:

`http.request.uri == https://www.wireshark.org/`

`http.host matches "acme\.(org|com|net)"`

`http.response.code == 200`

`http.request.method == "GET"`

`tcp contains "admin"`

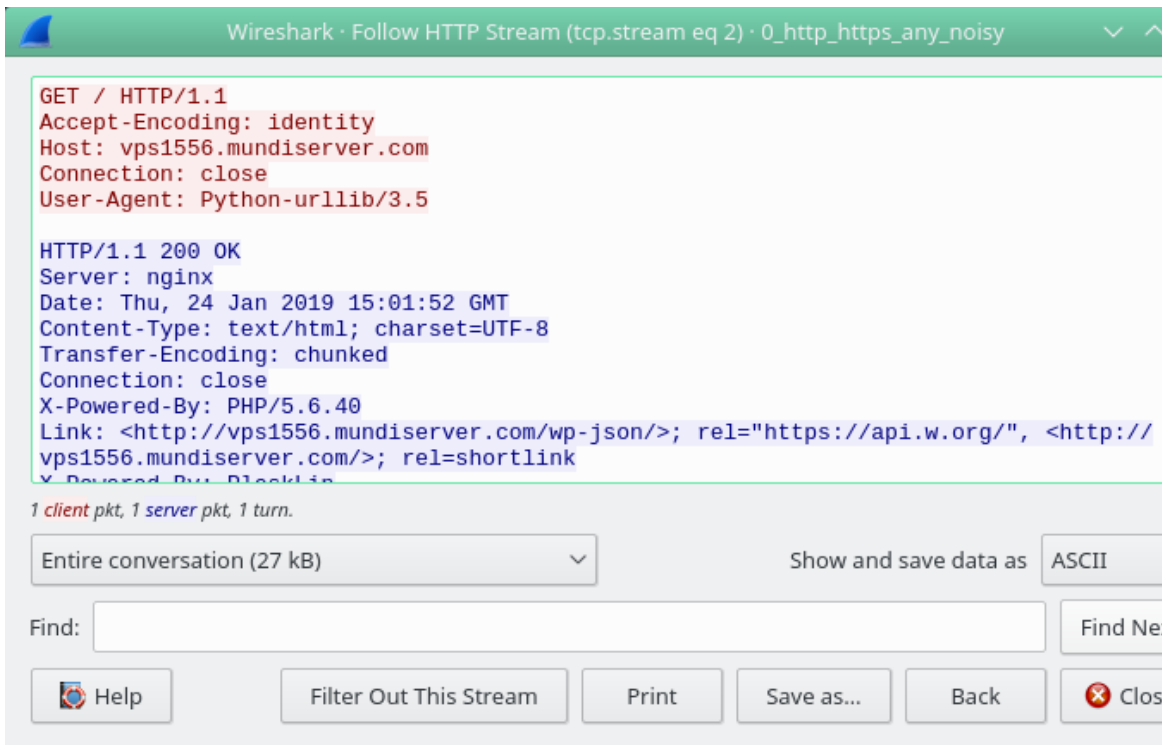


Filter for http traffic with specific addresses and frame time and not 200 response (e.g. you want to see 301 Moved permanently and 500 Server error packets):

```
http && ( (ip.dst == 192.168.178.27 ) || (ip.dst == 193.70.91.56 ) ) && frame.time > "2019-01-24 00:01:00.0000" && frame.time < "2019-01-25 15:01:53.0000" && http.response.code != 200
```

# Follow the stream

Select a Packet > Right mouse click > Follow > HTTP Stream



Wireshark · Follow HTTP Stream (tcp.stream eq 2) · 0\_http\_https\_any\_noisy

```
GET / HTTP/1.1
Accept-Encoding: identity
Host: vps1556.mundiserver.com
Connection: close
User-Agent: Python-urllib/3.5

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 24 Jan 2019 15:01:52 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.6.40
Link: <http://vps1556.mundiserver.com/wp-json/>; rel="https://api.w.org/", <http://vps1556.mundiserver.com/>; rel=shortlink
X-Powered-By: Blacklight
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (27 kB) Show and save data as ASCII

Find:  Find Next

Help Filter Out This Stream Print Save as... Back Close

# What if I told you...

---

That you can sort the traffic just by clicking the column names...

That you can search for strings in packets using Edit > Find Packet... (Ctrl+F)

- 
- SSL ManInTheMiddle with Wireshark (Linux edition)

To test the decryption of SSL traffic with Wireshark:

- Create private keys of the server and the client
- Start a server which uses the certificate with the key and send some test packets
- Configure Wireshark

# Create certificates

---

## Create a server certificate

```
# openssl req -new -x509 -out server.crt -nodes  
-keyout server.pem -subj /CN=localhost
```

## Create a client certificate

```
# openssl req -new -x509 -nodes -out client.crt  
-keyout client.key -subj /CN=Moi/O=Foo/C=NL
```

# Start a server

---

Start a server at localhost:4443

```
# openssl s_server -cipher AES256-SHA -accept 4443 -  
www -CAfile client.crt -verify 1 -key server.pem -  
cert server.crt
```



```
s_server -cipher AES256-SHA -accept 4443 -www -CAfile client.crt -verify 1 -key server.pem -cert server.crt
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:AES256-SHA
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-SHA          ECDHE-RSA-AES256-SHA
AES128-GCM-SHA256          AES256-GCM-SHA384          AES128-SHA
AES256-SHA                 DES-CBC3-SHA
Signature Algorithms: ECDSA+SHA256:0x04+0x08:RSA+SHA256:ECDSA+SHA384:0x05+0x08:RSA+SHA384:0x06+0x08:RSA+SHA512:RSA+SHA1
Shared Signature Algorithms: ECDSA+SHA256:RSA+SHA256:ECDSA+SHA384:RSA+SHA384:RSA+SHA512:RSA+SHA1
Supported Elliptic Curves: 0x8A8A:0x001D:P-256:P-384
Shared Elliptic curves: P-256:P-384
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
SSL-Session:
  Protocol   : TLSv1.2
  Cipher    : AES256-SHA
  Session-ID:
  Session-ID-ctx: 01000000
  Master-Key: C068D52572ABA77965042CA2E3E4BABA0474F7C45DE563C1226DFC2201AFEDF55BBF500A68FF48D260EE9DCB32BA59CA
  Key-Arg   : None
  PSK identity: None
  PSK identity hint: None
  SRP username: None
  Start Time: 1548434883
  Timeout   : 300 (sec)
  Verify return code: 0 (ok)
---
  0 items in the session cache
  0 client connects (SSL_connect())
```



## Send a request with python(3) and stop the capture

---

```
import urllib.request
import ssl

context = ssl._create_unverified_context()
with
urllib.request.urlopen("https://localhost:4443/",
context=context) as url:
    s = url.read()
    print(s)
```

# Traffic captured



No.	Time	Source	Destination	Protocol	Length	Info
22	5.110731826	IntelCor_58:61:64		ARP	44	192.168.178.27 is at b8:08:cf:58:61:64
23	5.362390056	127.0.0.1	127.0.0.1	TCP	76	56052 → 4443 [SYN] Seq=0 Win=43690 Len=0 MSS
24	5.362398184	127.0.0.1	127.0.0.1	TCP	76	4443 → 56052 [SYN, ACK] Seq=0 Ack=1 Win=4369
25	5.362405598	127.0.0.1	127.0.0.1	TCP	68	56052 → 4443 [ACK] Seq=1 Ack=1 Win=43776 Len
26	5.362506833	127.0.0.1	127.0.0.1	TLSv1.2	585	Client Hello
27	5.362518970	127.0.0.1	127.0.0.1	TCP	68	4443 → 56052 [ACK] Seq=1 Ack=518 Win=44800 L
28	5.362599242	127.0.0.1	127.0.0.1	TLSv1.2	1009	Server Hello, Certificate, Certificate Reque
29	5.362606016	127.0.0.1	127.0.0.1	TCP	68	56052 → 4443 [ACK] Seq=518 Ack=942 Win=45696
30	5.362841637	127.0.0.1	127.0.0.1	TLSv1.2	422	Certificate, Client Key Exchange, Change Cip
31	5.363809929	127.0.0.1	127.0.0.1	TLSv1.2	334	New Session Ticket, Change Cipher Spec, Fini
32	5.363964990	127.0.0.1	127.0.0.1	HTTP	233	GET / HTTP/1.1
33	5.364140080	127.0.0.1	127.0.0.1	TLSv1.2	3993	[SSL segment of a reassembled PDU]

# Configure wireshark

---

Edit > Preferences

On the left: Protocols > SSL

RSA keys list: press „Edit...“ and add via „+“

IP address – any

Port – 4443

Protocol – http

Key file – /.../server.pem

Password –

# Configure wireshark

context

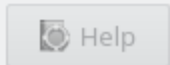
SSL Decrypt



IP address	Port	Protocol	Key File	Password
<input type="checkbox"/> any	4443	http	/data/home/Desktop/conf/owasp_wireshark/1/server.pem	



[/data/home/.config/wireshark/ssl\\_keys](/data/home/.config/wireshark/ssl_keys)



**SSL debug file (file with decrypted output):**

`/.../wiresharklog.txt`

Check „Reassemble SSL records spanning multiple TCP segments“

Check „Reassemble SSL Application Data spanning multiple SSL records“

Don't check “Message Authentication Code (MAC), ignore “mac failed”

Pre-Shared-Key (left empty):

**(Pre)-Master-Secret log filename:**

`/.../client.key`

Co

SSL  
/.../

Che  
Che  
Don

Pre-

(Pre  
/.../

ext

- SMPP
- SMTP
- SNA
- SNMP
- SoulSeek
- SoupBinTCP
- SPDY
- SPRT
- SRVLOC
- SSCOP
- SSH
- SSL**
- STANAG 5066 DTS
- STANAG 5066 SIS
- StarTeam
- STP

### Secure Sockets Layer

RSA keys list [Edit...](#)

#### SSL debug file

[Browse...](#)

- Reassemble SSL records spanning multiple TCP segments
- Reassemble SSL Application Data spanning multiple SSL records
- Message Authentication Code (MAC), ignore "mac failed"

Pre-Shared-Key

#### (Pre)-Master-Secret log filename

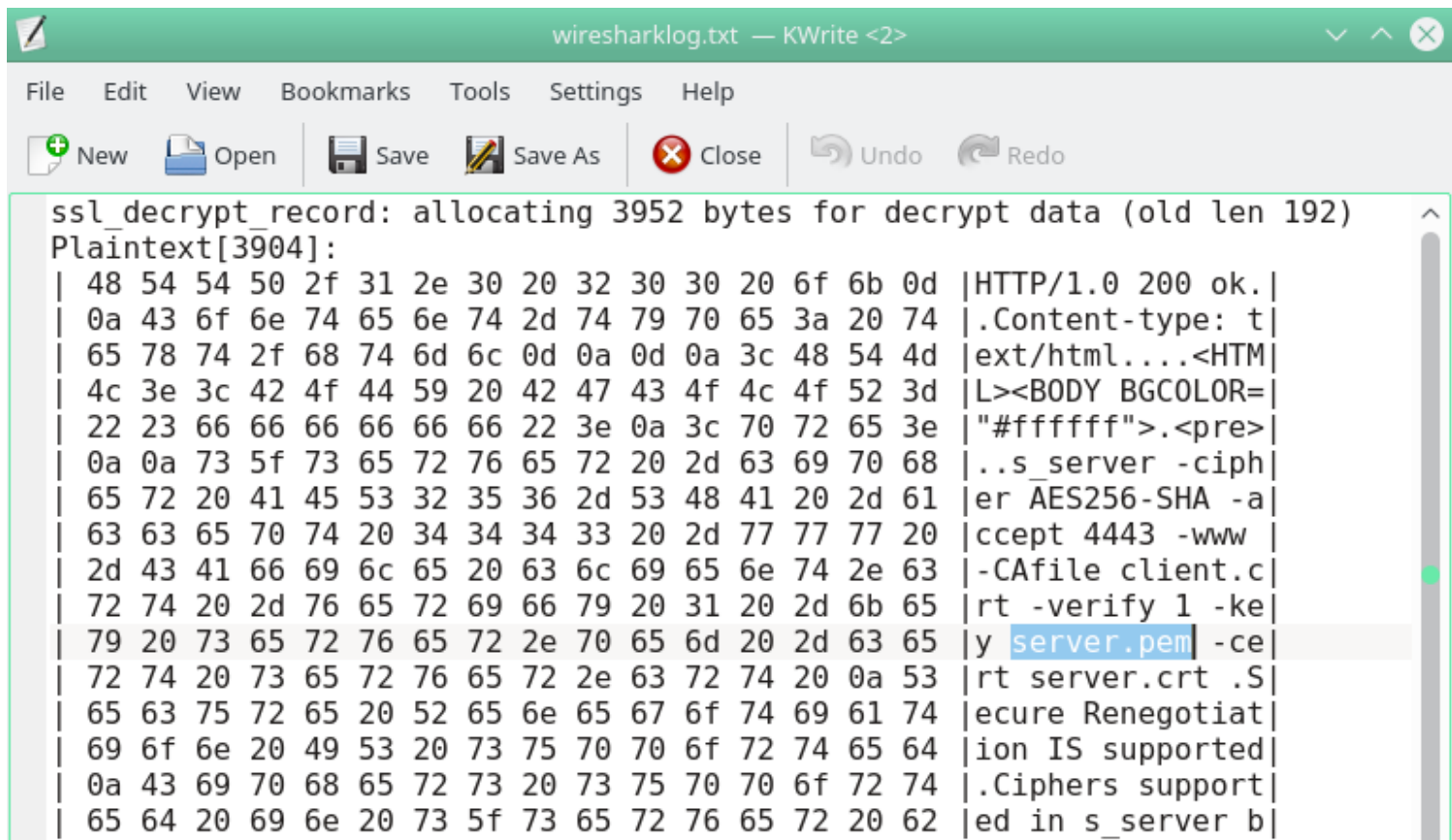
[Browse...](#)

Help

OK

Cancel

# Enjoy the decryption



The screenshot shows a KWrite window titled "wiresharklog.txt — KWrite <2>". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Tools", "Settings", and "Help". Below the menu bar is a toolbar with icons for "New", "Open", "Save", "Save As", "Close", "Undo", and "Redo". The main text area contains the following text:

```
ssl_decrypt_record: allocating 3952 bytes for decrypt data (old len 192)
Plaintext[3904]:
| 48 54 54 50 2f 31 2e 30 20 32 30 30 20 6f 6b 0d |HTTP/1.0 200 ok.|
| 0a 43 6f 6e 74 65 6e 74 2d 74 79 70 65 3a 20 74 |.Content-type: t|
| 65 78 74 2f 68 74 6d 6c 0d 0a 0d 0a 3c 48 54 4d |ext/html....<HTM|
| 4c 3e 3c 42 4f 44 59 20 42 47 43 4f 4c 4f 52 3d |L><BODY BGCOLOR=|
| 22 23 66 66 66 66 66 66 22 3e 0a 3c 70 72 65 3e |"#ffffff">.<pre>|
| 0a 0a 73 5f 73 65 72 76 65 72 20 2d 63 69 70 68 |..s_server -ciph|
| 65 72 20 41 45 53 32 35 36 2d 53 48 41 20 2d 61 |er AES256-SHA -a|
| 63 63 65 70 74 20 34 34 34 33 20 2d 77 77 77 20 |ccept 4443 -ww|
| 2d 43 41 66 69 6c 65 20 63 6c 69 65 6e 74 2e 63 |-CAfile client.c|
| 72 74 20 2d 76 65 72 69 66 79 20 31 20 2d 6b 65 |rt -verify 1 -ke|
| 79 20 73 65 72 76 65 72 2e 70 65 6d 20 2d 63 65 |y server.pem -ce|
| 72 74 20 73 65 72 76 65 72 2e 63 72 74 20 0a 53 |rt server.crt .S|
| 65 63 75 72 65 20 52 65 6e 65 67 6f 74 69 61 74 |ecure Renegotiat|
| 69 6f 6e 20 49 53 20 73 75 70 70 6f 72 74 65 64 |ion IS supported|
| 0a 43 69 70 68 65 72 73 20 73 75 70 70 6f 72 74 |.Ciphers support|
| 65 64 20 69 6e 20 73 5f 73 65 72 76 65 72 20 62 |ed in s_server b|
```

# Enjoy the decryption (proof)



← → ↻ ⚠ Not secure | https://localhost:4443

```
s_server -cipher AES256-SHA -accept 4443 -www -CAfile client.crt -verify 1 -key server.pem -cert server.crt
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:AES256-SHA
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-SHA ECDHE-RSA-AES256-SHA
AES128-GCM-SHA256 AES256-GCM-SHA384 AES128-SHA
AES256-SHA DES-CBC3-SHA
Signature Algorithms: ECDSA+SHA256:0x04+0x08:RSA+SHA256:ECDSA+SHA384:0x05+0x08:RSA+SHA384:0x06+0x08:RSA+SHA512:RSA+SHA512
Shared Signature Algorithms: ECDSA+SHA256:RSA+SHA256:ECDSA+SHA384:RSA+SHA384:RSA+SHA512:RSA+SHA1
Supported Elliptic Curves: 0x8A8A:0x001D:P-256:P-384
Shared Elliptic curves: P-256:P-384
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
SSL-Session:
  Protocol : TLSv1.2
  Cipher : AES256-SHA
  Session-ID:
  Session-ID-ctx: 01000000
  Master-Key: C068D52572ABA77965042CA2E3E4BABA0474F7C45DE563C1226DFC2201AFEDF55BBF500A68FF48D260EE9DCB32BA59CA
  Key-Arg : None
  PSK identity: None
  PSK identity hint: None
---
```



---

Set the `SSLKEYLOGFILE` environment variable and enter it's value under (Pre)-Master-Secret log filename.

<https://jimshaver.net/2015/02/11/decrypting-tls-browser-traffic-with-wireshark-the-easy-way/>

---

<https://wiki.wireshark.org/SSL>

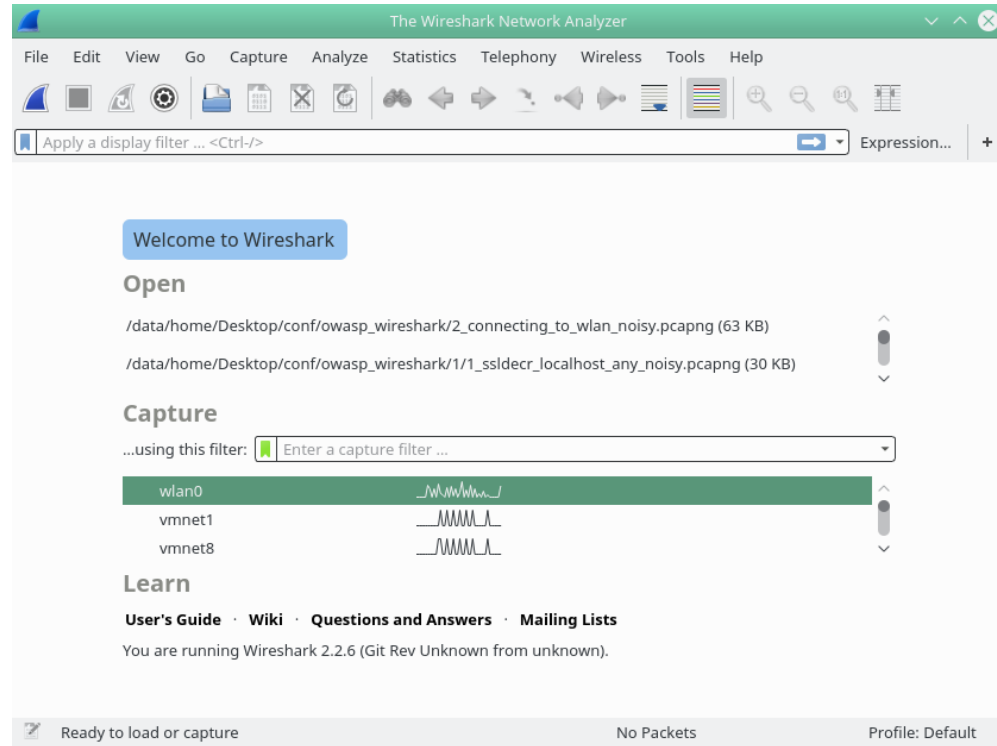
<https://www.cellstream.com/reference-reading/tipsandtricks/354-wireshark-ssl-tls-decryption>

- 
- WLAN traffic ManInTheMiddle with Wireshark

# Data packets capturing

To start capturing

- Select the **WLAN** network interface
- Click on the blue shark fin button / press Ctrl + E



# Example: Establishing a WLAN connection



No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.000000000	Avm_64:00:8f	IntelCor_58:61:64	EAPOL	113	Key (Message 1 of 4)
2	0.001357379	IntelCor_58:61:64	Avm_64:00:8f	EAPOL	135	Key (Message 2 of 4)
3	0.011014769	Avm_64:00:8f	IntelCor_58:61:64	EAPOL	185	Key (Message 3 of 4)
4	0.011116914	IntelCor_58:61:64	Avm_64:00:8f	EAPOL	113	Key (Message 4 of 4)
5	0.031731507	::	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
6	0.047907511	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x67554
7	0.055516718	192.168.178.1	192.168.178.27	DHCP	590	DHCP ACK - Transaction ID 0x67554
8	0.125767086	IntelCor_58:61:64	Broadcast	ARP	42	Who has 192.168.178.1? Tell 192.168.178.1
9	0.127793346	Avm_64:00:8d	IntelCor_58:61:64	ARP	42	192.168.178.1 is at 9c:c7:a6:64:00:8d
10	0.127806760	192.168.178.27	192.168.178.1	DNS	74	Standard query 0xeec5 A ntp.ubuntu.com
11	0.127818827	192.168.178.27	192.168.178.1	DNS	74	Standard query 0x78d8 AAAA ntp.ubuntu.com
12	0.131610265	192.168.178.1	192.168.178.27	DNS	138	Standard query response 0xeec5 A ntp.ubuntu.com
13	0.132247067	192.168.178.1	192.168.178.27	DNS	130	Standard query response 0x78d8 AAAA ntp.ubuntu.com

# Example: HTTP/HTTPS traffic capture on wlan0 interface



No.	Time	Source	Destination	Protocol	Lengt	Info
28	1.930272012	192.168.178.27	193.70.91.56	TCP	66	59584 → 80 [ACK] Seq=127 Ack=17281 Win=63872 Len=0 TSval=1872440954 ...
29	1.930298827	192.168.178.27	193.70.91.56	TCP	66	59584 → 80 [ACK] Seq=127 Ack=21601 Win=72448 Len=0 TSval=1872440954 ...
30	1.931667414	193.70.91.56	192.168.178.27	TCP	4386	[TCP segment of a reassembled PDU]
31	1.931784273	192.168.178.27	193.70.91.56	TCP	66	59584 → 80 [ACK] Seq=127 Ack=25921 Win=81152 Len=0 TSval=1872440954 ...
32	1.933904612	193.70.91.56	192.168.178.27	HTTP	1036	HTTP/1.1 200 OK (text/html)
33	1.934119278	192.168.178.27	193.70.91.56	TCP	66	59584 → 80 [FIN, ACK] Seq=127 Ack=26892 Win=83968 Len=0 TSval=187244...
34	1.948240022	193.70.91.56	192.168.178.27	TCP	66	80 → 59584 [ACK] Seq=26892 Ack=128 Win=14592 Len=0 TSval=4014201191 ...
35	3.942031300	192.168.178.27	192.168.178.1	DNS	68	Standard query 0x56fe A heise.de
36	3.942092098	192.168.178.27	192.168.178.1	DNS	68	Standard query 0x6b64 AAAA heise.de
37	3.946274849	192.168.178.1	192.168.178.27	DNS	84	Standard query response 0x56fe A heise.de A 193.99.144.80
38	3.947100931	192.168.178.1	192.168.178.27	DNS	96	Standard query response 0x6b64 AAAA heise.de AAAA 2a02:2e0:3fe:1001:...
39	3.947675431	192.168.178.27	193.99.144.80	TCP	74	59502 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1...
40	3.963331883	193.99.144.80	192.168.178.27	TCP	74	443 → 59502 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1452 SACK_PER...
41	3.963382014	192.168.178.27	193.99.144.80	TCP	66	59502 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1998354410 TSecr...
42	3.963761021	192.168.178.27	193.99.144.80	TLSv1.2	583	Client Hello
43	3.975707904	193.99.144.80	192.168.178.27	TCP	66	443 → 59502 [ACK] Seq=1 Ack=518 Win=30080 Len=0 TSval=413458293 TSecr...
44	3.978820431	193.99.144.80	192.168.178.27	TLSv1.2	3036	Server Hello, Certificate, Server Key Exchange, Server Hello Done
45	3.978947424	192.168.178.27	193.99.144.80	TCP	66	59502 → 443 [ACK] Seq=518 Ack=2971 Win=35200 Len=0 TSval=1998354414 ...
46	3.980880920	192.168.178.27	193.99.144.80	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

```
> HTTP/1.1 200 OK\r\n- Server: nginx\r\n- Date: Sun, 27 Jan 2019 14:19:42 GMT\r\n- Content-Type: text/html; charset=UTF-8\r\n- Transfer-Encoding: chunked\r\n- Connection: close\r\n
```

# Decrypt traffic with a known key

---

Edit > Preferences

On the left: Protocols > IEEE 802.11

And add a decryption key

- HDFSDATA
- HiSLIP
- HNBP
- HP\_ERM
- HPFEEDS
- HTTP
- IB
- ICEP
- ICMP
- IEEE 802.11**
- IEEE 802.15.4
- IEEE 802.1AH
- iFCP
- ILP
- IMAP
- IMF
- INAP

**IEEE 802.11 wireless LAN**

- Reassemble fragmented 802.11 datagrams
- Ignore vendor-specific HT elements
- Call subdissector for retransmitted 802.11 frames
- Assume packets have FCS
- Validate the FCS checksum if possible

## Ignore the Protection bit

- No
- Yes - without IV
- Yes - with IV

- Enable decryption

Key examples: 01:02:03:04:05 (40/64-bit WEP),  
010203040506070809101111213 (104/128-bit WEP),  
MyPassword[:MyAP] (WPA + plaintext password [+ SSID]),  
0102030405...6061626364 (WPA + 256-bit key). Invalid keys will be ignored.

Decryption Keys

[Edit...](#)

Help

OK

Cancel



## Aircrack-ng

<https://www.aircrack-ng.org/>

<https://tools.kali.org/wireless-attacks/aircrack-ng>

## EAPOL

<https://security.stackexchange.com/questions/66008/how-exactly-does-4-way-handshake-cracking-work>

# Wireshark Advanced

41 4C504F

- 
- Wireshark parsers (dissectors)

Dissectors are parsers/custom scripts to analyze packets' data.

Can be implemented

- In Lua language
- In C language

helloworld.lua (saved under /usr/lib/x86\_64-linux-gnu/wireshark/plugins/2.2.6/test/helloworld.lua):

```
local splash = TextWindow.new("Hello World!");
```

The dissector will be executed on Wireshark's start. The script has to be saved in the Plugin directory in this case, e.g.

```
# locate wireshark | grep -iE 'plugins'  
/usr/lib/x86_64-linux-gnu/wireshark/plugins/2.2.6
```

Alternatively, you can enforce the execution of a Lua dissector by running the dofile command under Tools > Lua > Evaluate

```
dofile("path/to/file.lua")
```

# Wireshark

The dissection  
has to be done  
# locate the  
/usr/lib

Alternative  
running

dofile(  
#

Wireshark · Evaluate Lua

```
dofile("/usr/lib/x86_64-linux-gnu/wireshark/plugins/2.2.6/test/  
helloworld.lua")
```

```
--[[ Evaluated --]]
```

Highlight:

Evaluate

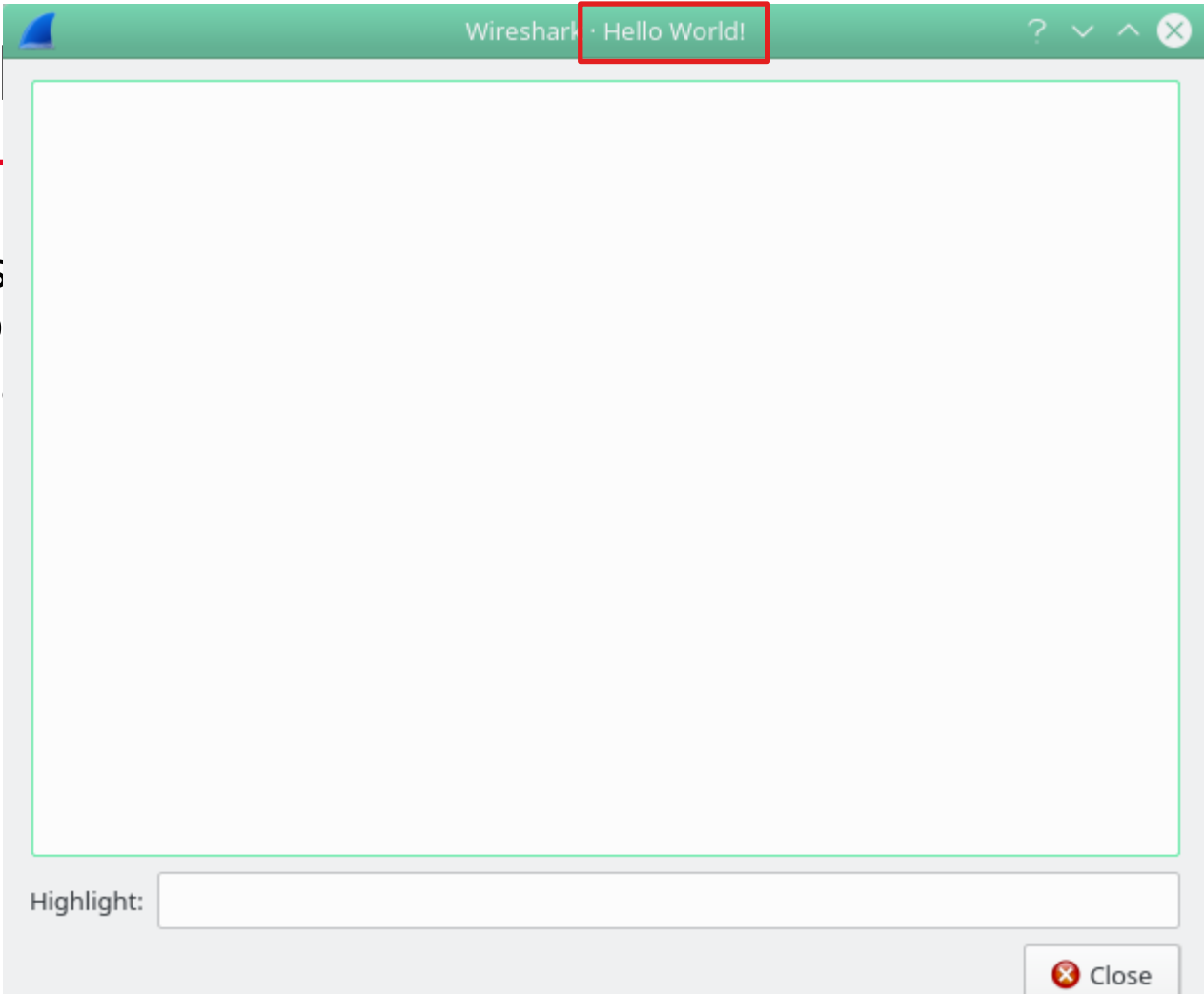
Close

context

script

sector by  
te

# Wireshark



context

The dissection  
has to be  
# located  
/usr/lib

script

Alternately  
running

sector by  
e

dofile(  
)



- Is a multi-paradigm language (supports procedural style, functional programming, has some object-oriented programming features)
- dynamically typed
- supports atomic data structures such as
  - boolean values,
  - numbers (double-precision floating point and 64-bit integers by default),
  - strings,
  - tables (for arrays/sets/lists)

- -- means comment
- Not equal in conditionals is `~=`
- Loops: while, repeat until (similar to a do while loop), for (numeric), for (generic).
- Use `i = i + 1` instead of `++` or `+=`
- nil for null

- Function example

```
function add(x, y)
    return x + y
```

```
end
```

```
local splash = TextWindow.new(add(3,6));
```

# Lua basics

context

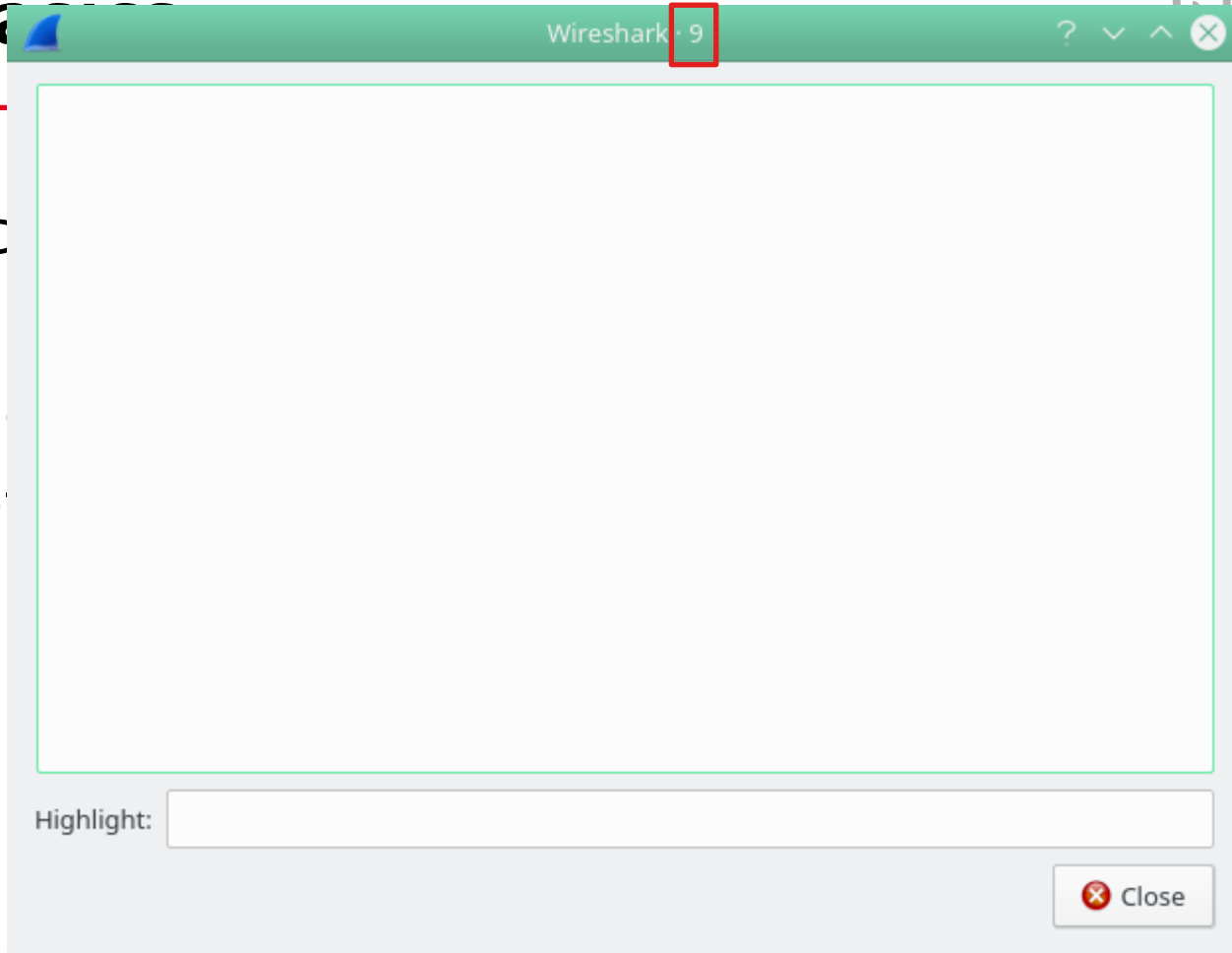
- Func

functi

re

end

local



# Lua basics (function example #2)



```
▼ function addto(x)
  -- Return a new function that adds x to the argument
▼ return function(y)
▼   --[=[ When we refer to the variable x, which is outside the current
      scope and whose lifetime would be shorter than that of this anonymous
      function, Lua creates a closure.]=]
      return x + y
    end
  end
fourplus = addto(4)
--This can also be achieved by calling the function in the following way:
print(addto(4)(3))
local splash = TextWindow.new(fourplus(3));
local splash = TextWindow.new(addto(4)(3));
```

# Lua b

context

```
function
  -- Ret
return
  -- [=

  retu
end
end
fourplus
--This c
print(ad
local sp
local sp
```

The image shows a screenshot of a Wireshark window. The window title bar is green and contains the text "Wireshark · 7 <2>". Below the title bar is a large, empty white rectangular area, likely a text editor or a display pane. At the bottom of the window, there is a "Highlight:" label followed by an empty text input field. In the bottom right corner of the window, there is a "Close" button with a red 'X' icon.

urrent  
is anonymous

ng way:

# Editing columns example

```
|- Append "<dst> -> <src>" to the Info column with a post-dissector.  
-- (Taps are not guaranteed to be run at a point when they can set the  
-- column text, so they can't be used for this.)  
  
-- create a new protocol so we can register a post-dissector  
local myproto = Proto("swapper", "Dummy proto to edit info column")  
  
-- the dissector function callback  
▼ function myproto.dissector(tvb, pinfo, tree)  
    pinfo.cols.info:append(" " .. tostring(pinfo.dst).. " -> "..tostring(pinfo.src))  
end  
-- register our new dummy protocol for post-dissection  
register_postdissector(myproto)
```

# Editing columns example (before lua)

 context

No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	38709 → 27017 [SYN] Seq=0 Win=32792
2	0.000041	127.0.0.1	127.0.0.1	TCP	74	27017 → 38709 [SYN, ACK] Seq=0 Ack=
3	0.000064	127.0.0.1	127.0.0.1	TCP	66	38709 → 27017 [ACK] Seq=1 Ack=1 Win
4	0.011117	127.0.0.1	127.0.0.1	MONGO	126	Request : Query
5	0.012592	127.0.0.1	127.0.0.1	TCP	66	27017 → 38709 [ACK] Seq=1 Ack=61 Wi
6	0.013028	127.0.0.1	127.0.0.1	MONGO	144	Response : Reply
7	0.013372	127.0.0.1	127.0.0.1	TCP	66	38709 → 27017 [ACK] Seq=61 Ack=79 W
8	3.703972	127.0.0.1	127.0.0.1	MONGO	148	Request : Insert document

> Frame 4: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)



# Editing columns example (after execution)



No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	38709 → 27017 [SYN] Seq=0 Win=32792 Len=0 M
2	0.000041	127.0.0.1	127.0.0.1	TCP	74	27017 → 38709 [SYN, ACK] Seq=0 Ack=1 Win=32
3	0.000064	127.0.0.1	127.0.0.1	TCP	66	38709 → 27017 [ACK] Seq=1 Ack=1 Win=32800 L
4	0.011117	127.0.0.1	127.0.0.1	MONGO	126	Request : Query 127.0.0.1 -> 127.0.0.1
5	0.012592	127.0.0.1	127.0.0.1	TCP	66	27017 → 38709 [ACK] Seq=1 Ack=61 Win=32768
6	0.013028	127.0.0.1	127.0.0.1	MONGO	144	Response : Reply 127.0.0.1 -> 127.0.0.1
7	0.013372	127.0.0.1	127.0.0.1	TCP	66	38709 → 27017 [ACK] Seq=61 Ack=79 Win=32800
8	3.703972	127.0.0.1	127.0.0.1	MONGO	148	Request : Insert document 127.0.0.1 -> 127.

> Frame 4: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)

Note: will only work at Wireshark's start (save the script in the Plugins folder before)

# Editing trees example

```
local proto_foo = Proto("foo", "Foo Protocol")
proto_foo.fields.bytes = ProtoField.bytes("foo.bytes", "Byte array")
proto_foo.fields.ul6 = ProtoField.uint16("foo.ul6", "Unsigned short", base.HEX)

function proto_foo.dissector(buf, pinfo, tree)
  -- ignore packets less than 4 bytes long
  if buf:len() < 4 then return end

  -- #####
  -- # Assume buf(0,4) == {0x00, 0x01, 0x00, 0x02}
  -- #####

  local t = tree:add( proto_foo, buf() )

  -- Adds a byte array that shows as: "Byte array: 00010002"
  t:add( proto_foo.fields.bytes, buf(0,4) )

  -- Adds a byte array that shows as "Byte array: 313233"
  -- (the ASCII char code of each character in "123")
  t:add( proto_foo.fields.bytes, buf(0,4), "123" )

  -- Adds a tree item that shows as: "Unsigned short: 0x0001"
  t:add( proto_foo.fields.ul6, buf(0,2) )
```

# Editing trees example

```
-- Adds a tree item that shows as: "Unsigned short: 0x0064"
t:add( proto_foo.fields.u16, buf(0,2), 100 )

-- Adds a tree item that shows as: "Unsigned short: 0x0064 ( big endian )"
t:add( proto_foo.fields.u16, buf(1,2), 100, nil, "(", nil, "big", 999, nil, "endian", nil, ")" )

-- LITTLE ENDIAN: Adds a tree item that shows as: "Unsigned short: 0x0100"
t:add_le( proto_foo.fields.u16, buf(0,2) )

-- LITTLE ENDIAN: Adds a tree item that shows as: "Unsigned short: 0x6400"
t:add_le( proto_foo.fields.u16, buf(0,2), 100 )

-- LITTLE ENDIAN: Adds a tree item that shows as: "Unsigned short: 0x6400 ( little endian )"
t:add_le( proto_foo.fields.u16, buf(1,2), 100, nil, "(", nil, "little", 999, nil, "endian", nil, ")" )

end

udp_table = DissectorTable.get("udp.port")
udp_table:add(32768, proto_foo)
```

# Editing trees example

```
10 0.158315      139.133.204.176      139.133.204.183      UDP-Lite      60 32768 → 1234 Len=12
>-Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
>-Ethernet II, Src: 3Com_c7:87:49 (00:04:75:c7:87:49), Dst: 3Com_dd:bb:3a (00:04:76:dd:bb:3a)
>-Internet Protocol Version 4, Src: 139.133.204.176, Dst: 139.133.204.183
>-Lightweight User Datagram Protocol, Src Port: 32768, Dst Port: 1234
v- Foo Protocol
  | Byte array: 68656c6c
  | Byte array: 31323300
  | Unsigned short: 0x6865
  | Unsigned short: 0x0064
  | Unsigned short: 0x0064 ( big 999 endian )
  | Unsigned short: 0x6568
  | Unsigned short: 0x0064
  | Unsigned short: 0x0064 ( little 999 endian )
0000  00 04 76 dd bb 3a 00 04 75 c7 87 49 08 00 45 00  ..v.... u..I..E.
0010  00 28 52 a6 40 00 40 88 37 35 8b 85 cc b0 8b 85  .(R.@.@. 75.....
0020  cc b7 80 00 04 d2 00 11 9c aa 68 65 6c 6c 6f 20  ..... ..hello
0030  77 6f 72 6c 64 0a 00 00 00 00 00 00          world... ..
```

# Editing trees example

```
10 0.158315      139.133.204.176      139.133.204.183      UDP-Lite      60 32768 → 1234 Len=12
```

---

```
>-Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
>-Ethernet II, Src: 3Com_c7:87:49 (00:04:75:c7:87:49), Dst: 3Com_dd:bb:3a (00:04:76:dd:bb:3a)
>-Internet Protocol Version 4, Src: 139.133.204.176, Dst: 139.133.204.183
>-Lightweight User Datagram Protocol, Src Port: 32768, Dst Port: 1234
v- Foo Protocol
  - Byte array: 68656c6c
  - Byte array: 31323300
  - Unsigned short: 0x6865
  - Unsigned short: 0x0064
  - Unsigned short: 0x0064 ( big 999 endian )
  - Unsigned short: 0x6568
  - Unsigned short: 0x0064
  - Unsigned short: 0x0064 ( little 999 endian )
```

---

```
0000  00 04 76 dd bb 3a 00 04 75 c7 87 49 08 00 45 00  ..v.... u..I..E.
0010  00 28 52 a6 40 00 40 88 37 35 8b 85 cc 00 8b 85  .(R.@.@. 75.....
0020  cc b7 80 00 04 d2 00 11 9c aa 68 65 6c 6c 6f 20  ..... ..hello
0030  77 6f 72 6c 64 0a 00 00 00 00 00 00  world... ..
```

---

Edit the script so that

- it works for HTTP protocol on port 80

Add a function

- e.g. addition of 2 values
- output the result in a tree field

# HTTP Example

```
local proto_foo = Proto("foo", "Foo Protocol")
proto_foo.fields.bytes = ProtoField.bytes("foo.bytes", "Byte array")
proto_foo.fields.u16 = ProtoField.uint16("foo.u16", "Unsigned short", base.HEX)
```

```
▼ function addxy(x, y)
    return x + y
end
```

```
▼ function proto_foo.dissector(buf, pinfo, tree)
    -- ignore packets less than 1 bytes long
    if buf:len() < 1 then return end

    local t = tree:add( proto_foo, buf() )

    t:add( proto_foo.fields.bytes, buf(), string.char(addxy(2,2)) )
    t:add( proto_foo.fields.bytes, buf(), string.char(addxy(2,3)) )

end
```

```
tcp_table = DissectorTable.get("tcp.port")
tcp_table:add(80, proto_foo)
```

# HTTP Example

62	1.302755805	193.70.91.56	10.49.3.71	TCP	1038 80 → 47502 [FIN, PSH, ACK] Seq=25921
63	1.302921253	10.49.3.71	193.70.91.56	TCP	68 47502 → 80 [FIN, ACK] Seq=127 Ack=25921
64	1.326787272	193.70.91.56	10.49.3.71	TCP	68 80 → 47502 [ACK] Seq=26892 Ack=127

```
>-Frame 62: 1038 bytes on wire (8304 bits), 1038 bytes captured (8304 bits) on interface 0
>-Linux cooked capture
>-Internet Protocol Version 4, Src: 193.70.91.56, Dst: 10.49.3.71
>-Transmission Control Protocol, Src Port: 80, Dst Port: 47502, Seq: 25921, Ack: 127, Len: 970
>-Foo Protocol
  -Byte array: 0400083e9cd550000300000000000000000910000000000000...
  -Byte array: 050069006c00650000000000cd550000e0e3fb742b7f0000...
```

0040	53 f3 13 89 2f 61 64 6d 69 6e 2d 61 6a 61 78 2e	S.../adm in-ajax.
0050	70 68 70 22 2c 22 77 63 5f 61 6a 61 78 5f 75 72	php", "wc _ajax_ur
0060	6c 22 3a 22 5c 2f 3f 77 63 2d 61 6a 61 78 3d 25	l": "\/?w c-ajax=%
0070	25 65 6e 64 70 6f 69 6e 74 25 25 22 2c 22 63 61	%endpoin t%", "ca
0080	72 74 5f 68 61 73 68 5f 6b 65 79 22 3a 22 77 63	rt_hash_ key": "wc
0090	5f 63 61 72 74 5f 68 61 73 68 5f 32 31 35 64 63	_cart_ha sh_215dc
00a0	31 63 61 62 31 31 64 64 64 34 35 34 31 65 62 63	1cab11dd d4541ebc
00b0	36 61 66 61 36 39 33 33 30 66 36 22 2c 22 66 72	6afa6933 0f6", "fr
00c0	61 67 6d 65 6e 74 5f 6e 61 6d 65 22 3a 22 77 63	agment_n ame": "wc
00d0	5f 66 72 61 67 6d 65 6e 74 73 5f 32 31 35 64 63	_fragmen ts_215dc
00e0	31 63 61 62 31 31 64 64 64 34 35 34 31 65 62 63	1cab11dd d4541ebc
00f0	36 61 66 61 36 39 33 33 30 66 36 22 7d 3b 0a 2f	6afa6933 0f6"}; /
0100	2a 20 5d 5d 2e 20 2a 2f 0a 3c 2f 73 63 72 69 70	* ]]> */ </scrip



- 
- Develop a dissector to encode the request body of a HTTP packet into the Base64 format (if you know how to encode it, you will probably be able to decode it;))

- Develop a dissector to encode the request body of a HTTP packet into the Base64 format (if you know how to encode it, you will probably be able to decode it;))

## Hints

<https://wiki.wireshark.org/LuaAPI/Treeltem>

<https://github.com/toastdriven/lua-base64/blob/master/base64.lua>

# Exercise (solution)

- Replace the `addxy` function with the code from <https://github.com/toastdriven/lua-base64/blob/master/base64.lua>
- Edit `proto_foo.dissector` as follows:

# Exercise (solution)

```
▼ function proto_foo.dissector(buf, pinfo, tree)
  -- ignore packets less than 1 bytes long
  if buf:len() < 1 then return end

  local t = tree:add( proto_foo, buf() )

  t:set_text( to_base64(tostring(buf())) )
  --t:set_text( from_base64(to_base64(tostring(buf()))) )

  --test with 'ab' string
  --t:set_text( to_base64('ab') )
end

tcp_table = DissectorTable.get("tcp.port")
tcp_table:add(80, proto_foo)
```

# Exercise (solution)

No.	Time	Source	Destination	Protocol	Length	Info
60	1.301657828	193.70.91.56	10.49.3.71	TCP	1508	80 → 47502 [ACK] Seq=24481 Ack=127
61	1.301701992	10.49.3.71	193.70.91.56	TCP	68	47502 → 80 [ACK] Seq=127 Ack=25921
62	1.302755805	193.70.91.56	10.49.3.71	TCP	1038	80 → 47502 [FIN, PSH, ACK] Seq=25921
63	1.302921253	10.49.3.71	193.70.91.56	TCP	68	47502 → 80 [FIN, ACK] Seq=127 Ack=25921
64	1.326787272	193.70.91.56	10.49.3.71	TCP	68	80 → 47502 [ACK] Seq=26892 Ack=127

```
> Frame 62: 1038 bytes on wire (8304 bits), 1038 bytes captured (8304 bits) on interface 0
> Linux cooked capture
> Internet Protocol Version 4, Src: 193.70.91.56, Dst: 10.49.3.71
> Transmission Control Protocol, Src Port: 80, Dst Port: 47502, Seq: 25921, Ack: 127, Len: 970
  MmY2MTY0NmQ2OTZlMmQ2MTZhNjE3ODJlNzA2ODcwMjIyYzIyNzc2MzVmNjE2YTlxLi4u
```

```
0040 53 f3 13 89 2f 61 64 6d 69 6e 2d 61 6a 61 78 2e S.../adm in-ajax.
0050 70 68 70 22 2c 22 77 63 5f 61 6a 61 78 5f 75 72 php", "wc _ajax_ur
0060 6c 22 3a 22 5c 2f 3f 77 63 2d 61 6a 61 78 3d 25 l": "\/?w c-ajax=%
0070 25 65 6e 64 70 6f 69 6e 74 25 25 22 2c 22 63 61 %endpoin t%", "ca
0080 72 74 5f 68 61 73 68 5f 6b 65 79 22 3a 22 77 63 rt_hash_ key": "wc
0090 5f 63 61 72 74 5f 68 61 73 68 5f 32 31 35 64 63 _cart_ha sh_215dc
00a0 31 63 61 62 31 31 64 64 64 34 35 34 31 65 62 63 1cab11dd d4541ebc
00b0 36 61 66 61 36 39 33 33 30 66 36 22 2c 22 66 72 6afa6933 0f6", "fr
00c0 61 67 6d 65 6e 74 5f 6e 61 6d 65 22 3a 22 77 63 agment_n ame": "wc
00d0 5f 66 72 61 67 6d 65 6e 74 73 5f 32 31 35 64 63 _fragmen ts_215dc
00e0 2f 62 61 62 21 21 64 64 64 24 25 24 21 65 62 62 1cab11dd d4541ebc
```

# Exercise (solution, proof)

- Decode as a proof

```
t:set_text(  
from_base64(to_base64(tostring(buf())))) )
```

# Exercise (solution)



62	1.302755805	193.70.91.56	10.49.3.71	TCP	1038	80 → 47502	[FIN, PSH, ACK] Seq=25
63	1.302921253	10.49.3.71	193.70.91.56	TCP	68	47502 → 80	[FIN, ACK] Seq=127 Ack
64	1.326787272	193.70.91.56	10.49.3.71	TCP	68	80 → 47502	[ACK] Seq=26892 Ack=12

```
>-Frame 62: 1038 bytes on wire (8304 bits), 1038 bytes captured (8304 bits) on interface 0
>-Linux cooked capture
>-Internet Protocol Version 4, Src: 193.70.91.56, Dst: 10.49.3.71
>-Transmission Control Protocol, Src Port: 80, Dst Port: 47502, Seq: 25921, Ack: 127, Len: 970
  2f61646d696e2d616a61782e706870222c2277635f616a61...
```

0040	53 f3 13 89	2f 61 64 6d	69 6e 2d 61	6a 61 78 2e	S.../adm in-ajax.
0050	70 68 70 22	2c 22 77 63	5f 61 6a 61	78 5f 75 72	php", "wc _ajax_ur
0060	6c 22 3a 22	5c 2f 3f 77	63 2d 61 6a	61 78 3d 25	l": "\/?w c-ajax=%
0070	25 65 6e 64	70 6f 69 6e	74 25 25 22	2c 22 63 61	%endpoin t%%", "ca
0080	72 74 5f 68	61 73 68 5f	6b 65 79 22	3a 22 77 63	rt_hash_ key": "wc
0090	5f 63 61 72	74 5f 68 61	73 68 5f 32	31 35 64 63	_cart_ha sh_215dc
00a0	31 63 61 62	31 31 64 64	64 34 35 34	31 65 62 63	1cab11dd d4541ebc
00b0	36 61 66 61	36 39 33 33	30 66 36 22	2c 22 66 72	6afa6933 0f6", "fr
00c0	61 67 6d 65	6e 74 5f 6e	61 6d 65 22	3a 22 77 63	agment_n ame": "wc
00d0	5f 66 72 61	67 6d 65 6e	74 73 5f 32	31 35 64 63	_fragmen ts_215dc
00e0	31 63 61 62	31 31 64 64	64 34 35 34	31 65 62 63	1cab11dd d4541ebc
00f0	36 61 66 61	36 39 33 33	30 66 36 22	7d 3b 0a 2f	6afa6933 0f6"};./
0100	2a 20 5d 5d	3e 20 2a 2f	0a 3c 2f 73	63 72 69 70	* ]]> */ ./scrip
0110	74 3e 0a 3c	73 63 72 69	70 74 20 74	79 70 65 3d	t>.<scri pt type=

# What's next?

---

 context



# What's next?

---

 context



# What's next?

---

## Modify/resend packets?

- > use Burp, OWASP ZAP etc. proxy for HTTP/HTTPS,
- > tcpreplay, tcprewrite, tcpreplay-edit
- > Canape (if you dare)

# Questions/Feedback?

**E-mail: [414C504F@tuta.io](mailto:414C504F@tuta.io)**

**Github: <https://github.com/414C504F>**

**Thanks!**