How to Verify Real User Monitor (RUM) SSL Keys

Here are the steps to verify if an SSL key is valid and could be used to decrypt application traffic. The actions, described below, should be performed at a customer side, that is really a secure way as the customer does not need to provide you with sensitive/private information.

Input data:

1. Captured application traffic. Could be obtained using one of next means:
   a. Capturing from Engine webconsole: Configuration => Probe management => Probe Traffic Capture. That's the easiest way.
   b. Using Wireshark on windows machine or tcpdump on linux. It is assumed that the traffic is delivered to the machine using either port mirroring or TAP device.
2. SSL key in unencrypted PEM format. As #PKCS12 format is used widely and it's supported by RUM, the customer often use it (*.pfx or *.p12 file extension). To convert from PFX/P12 to unprotected PEM use next command

   ```
   openssl pkcs12 -in inputKey.pfx -out outputKey.pem –nodes –nocerts
   ```

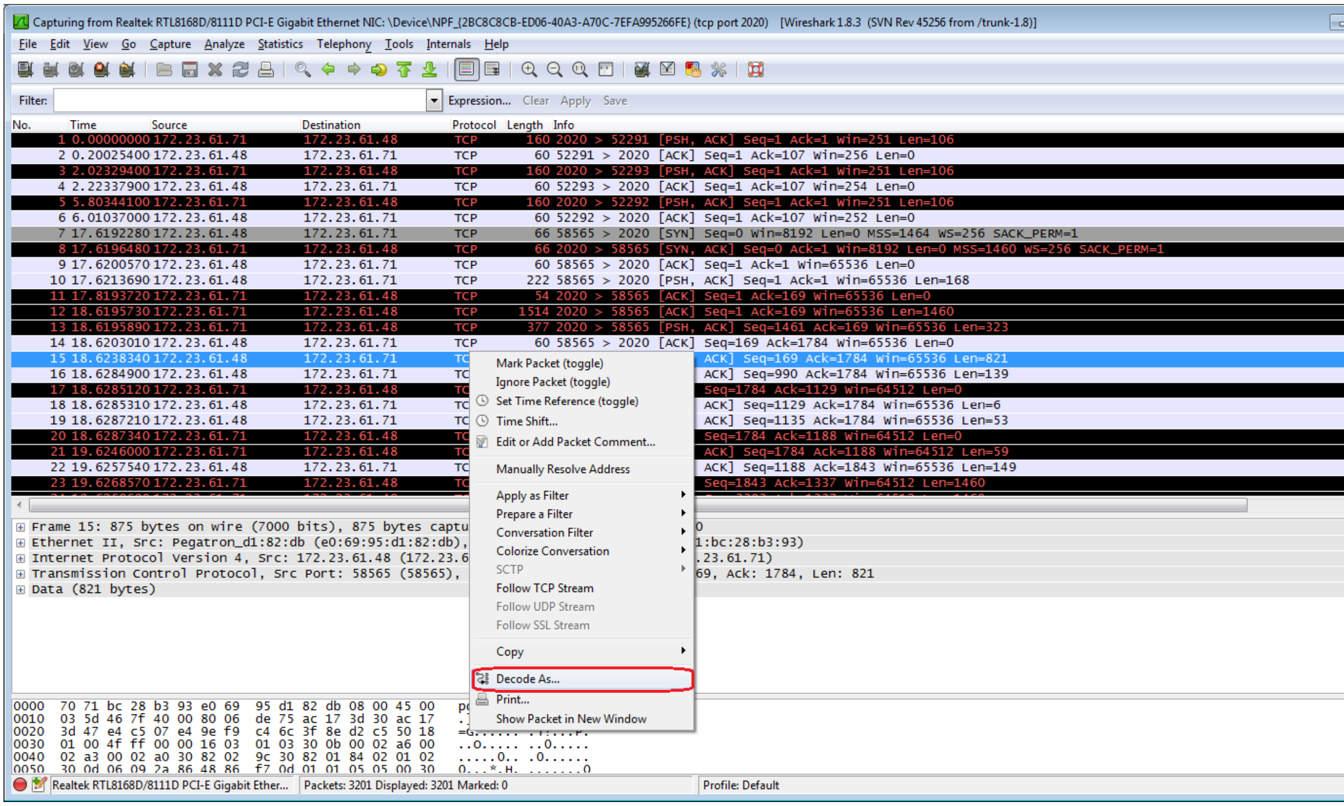
   You will be prompted to input password, which should be provided by customer.
   See http://www.openssl.org/docs/apps/pkcs12.html or http://jefferytay.wordpress.com/2010/12/09/converting-a-pfx-file-to-pem-and-key-via-openssl/ or google for more details.
   Verify that the resulting outputKey.pem does not contain 'Proc-Type: 4,ENCRYPTED' string.

Tools:

- Wireshark which is a free tool for traffic sniffing and analyzing http://www.wireshark.org/download.html
- Openssl to convert private key to unprotected PEM format. Windows installer can be found at http://slproweb.com/products/Win32OpenSSL.html

Steps:

1. Open the captured traffic file in the Wireshark
2. [optional step, for non default https server port only] If the customer's web/application server uses different port than 443, you should tell the Wireshark about that: Right click on a packet and select 'Decode as'



Capturing from Realtek RTL8168D/8111D PCI-E Gigabit Ethernet NIC: \Device\NPF_{2BC8C8CB-ED06-40A3-A70C-7EFA995266FE} (tcp port 2020) [Wireshark 1.8.3 (SVN Rev 45256 from /trunk-1.8)]

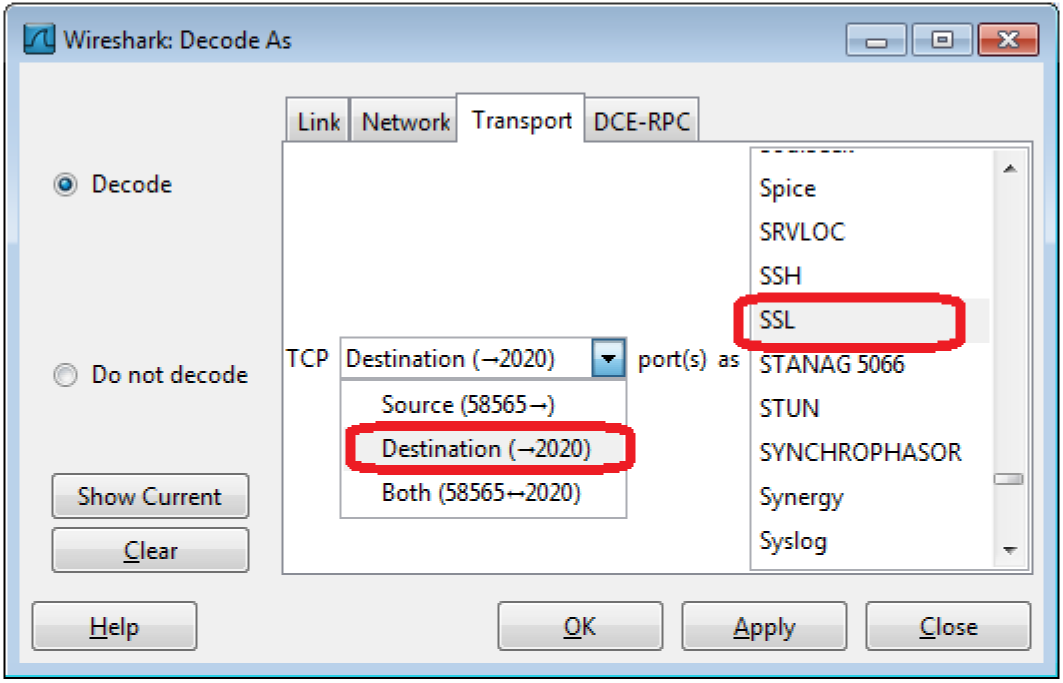
```
No.  Time        Source        Destination   Protocol Length Info
  1 0.00000000  172.23.61.71  172.23.61.48  TCP      160 2020 > 52291 [PSH, ACK] Seq=1 Ack=1 Win=251 Len=106
  2 0.20025400  172.23.61.48  172.23.61.71  TCP       60 52291 > 2020 [ACK] Seq=1 Ack=107 Win=256 Len=0
  3 2.02329400  172.23.61.71  172.23.61.48  TCP      160 2020 > 52293 [PSH, ACK] Seq=1 Ack=1 Win=251 Len=106
  4 2.22337900  172.23.61.48  172.23.61.71  TCP       60 52293 > 2020 [ACK] Seq=1 Ack=107 Win=254 Len=0
  5 5.80344100  172.23.61.71  172.23.61.48  TCP      160 2020 > 52292 [PSH, ACK] Seq=1 Ack=1 Win=251 Len=106
  6 6.01037000  172.23.61.48  172.23.61.71  TCP       60 52292 > 2020 [ACK] Seq=1 Ack=107 Win=252 Len=0
  7 17.6192280  172.23.61.48  172.23.61.71  TCP       66 58565 > 2020 [SYN] Seq=0 Win=8192 Len=0 MSS=1464 WS=256 SACK_PERM=1
  8 17.6196480  172.23.61.71  172.23.61.48  TCP       66 2020 > 58565 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  9 17.6200570  172.23.61.48  172.23.61.71  TCP       60 58565 > 2020 [ACK] Seq=1 Ack=1 Win=65536 Len=0
 10 17.6213690  172.23.61.48  172.23.61.71  TCP      222 58565 > 2020 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=168
 11 17.8193720  172.23.61.71  172.23.61.48  TCP       54 2020 > 58565 [ACK] Seq=1 Ack=169 Win=65536 Len=0
 12 18.6195730  172.23.61.71  172.23.61.48  TCP     1514 2020 > 58565 [ACK] Seq=1 Ack=169 Win=65536 Len=1460
 13 18.6195890  172.23.61.71  172.23.61.48  TCP      377 2020 > 58565 [PSH, ACK] Seq=1461 Ack=169 Win=65536 Len=323
 14 18.6203010  172.23.61.48  172.23.61.71  TCP       60 58565 > 2020 [ACK] Seq=169 Ack=1784 Win=65536 Len=0
 15 18.6238340  172.23.61.48  172.23.61.71  TC        ACK] Seq=169 Ack=1784 Win=65536 Len=821
 16 18.6284900  172.23.61.48  172.23.61.71  TC        ACK] Seq=990 Ack=1784 Win=65536 Len=139
 17 18.6285120  172.23.61.71  172.23.61.48  TC        Seq=1784 Ack=1129 Win=64512 Len=0
 18 18.6285310  172.23.61.48  172.23.61.71  TC        ACK] Seq=1129 Ack=1784 Win=65536 Len=6
 19 18.6287210  172.23.61.48  172.23.61.71  TC        ACK] Seq=1135 Ack=1784 Win=65536 Len=53
 20 18.6287340  172.23.61.71  172.23.61.48  TC        Seq=1784 Ack=1188 Win=64512 Len=0
 21 19.6246000  172.23.61.71  172.23.61.48  TC        ACK] Seq=1784 Ack=1188 Win=64512 Len=59
 22 19.6257540  172.23.61.48  172.23.61.71  TC        ACK] Seq=1188 Ack=1843 Win=65536 Len=149
 23 19.6268570  172.23.61.71  172.23.61.48  TC        Seq=1843 Ack=1337 Win=64512 Len=1460
```

```
⊞ Frame 15: 875 bytes on wire (7000 bits), 875 bytes captu                          0
⊞ Ethernet II, Src: Pegatron_d1:82:db (e0:69:95:d1:82:db),                          1:bc:28:b3:93)
⊞ Internet Protocol Version 4, Src: 172.23.61.48 (172.23.6                          .23.61.71)
⊞ Transmission Control Protocol, Src Port: 58565 (58565),                           69, Ack: 1784, Len: 821
⊞ Data (821 bytes)
```

```
0000  70 71 bc 28 b3 93 e0 69  95 d1 82 db 08 00 45 00   pq
0010  03 5d 46 7f 40 00 80 06  de 75 ac 17 3d 30 ac 17   .]
0020  3d 47 e4 c5 07 e4 9e f9  c4 6c 3f 8e d2 c5 50 18   =G...... .?....P.
0030  01 00 4f ff 00 00 16 03  01 03 30 0b 00 02 a6 00   ..O..... ..0.....
0040  02 a3 00 02 a0 30 82 02  9c 30 82 01 84 02 01 02   .....0.. .0......
0050  30 0d 06 09 2a 86 48 86  f7 0d 01 01 05 05 00 30   0...*.H. .......0
```

Realtek RTL8168D/8111D PCI-E Gigabit Ether... | Packets: 3201 Displayed: 3201 Marked: 0 | Profile: Default

Select 'Transport' tab and server port. On the picture below we tell the Wireshark that traffic with port 2020 is SSL encrypted



Wireshark: Decode As

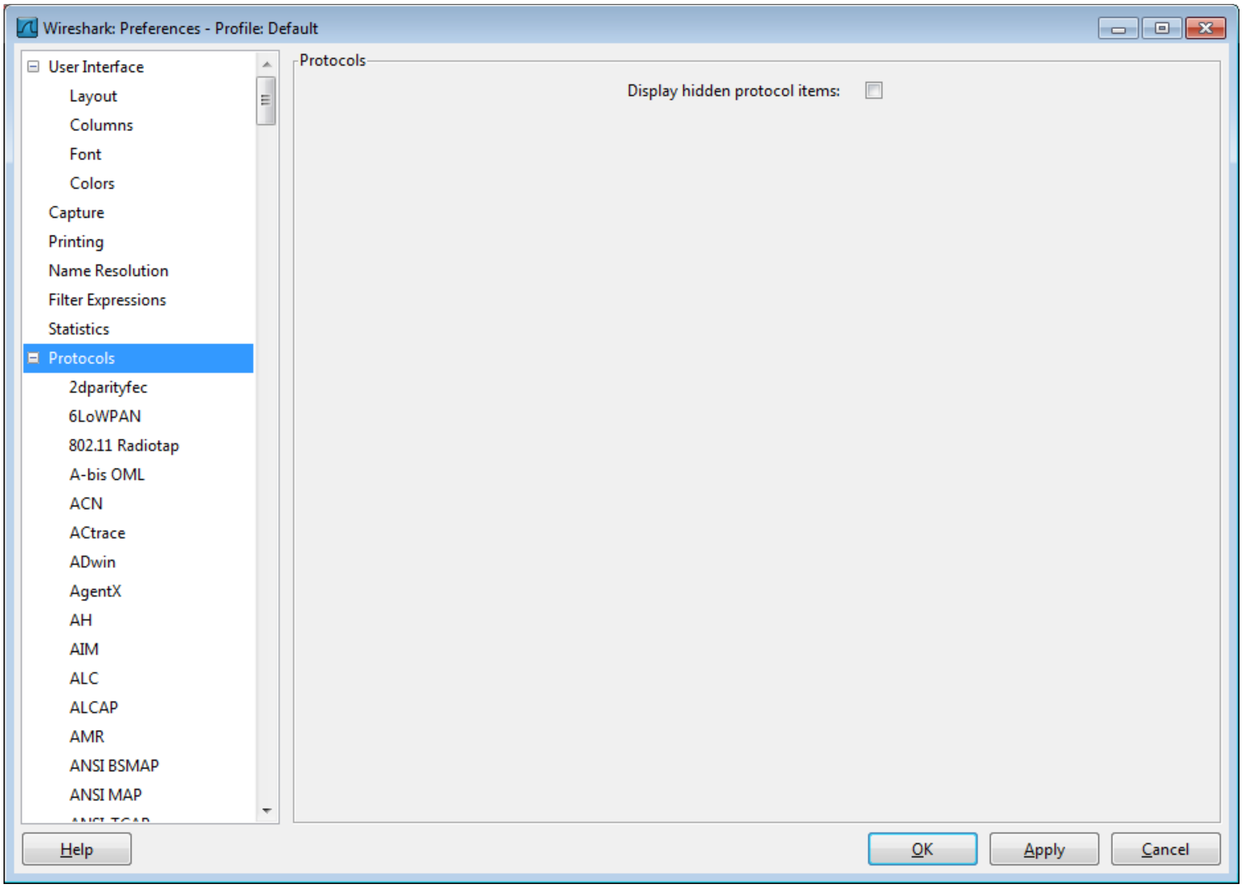
● Decode

○ Do not decode

TCP Destination (→2020) port(s) as SSL

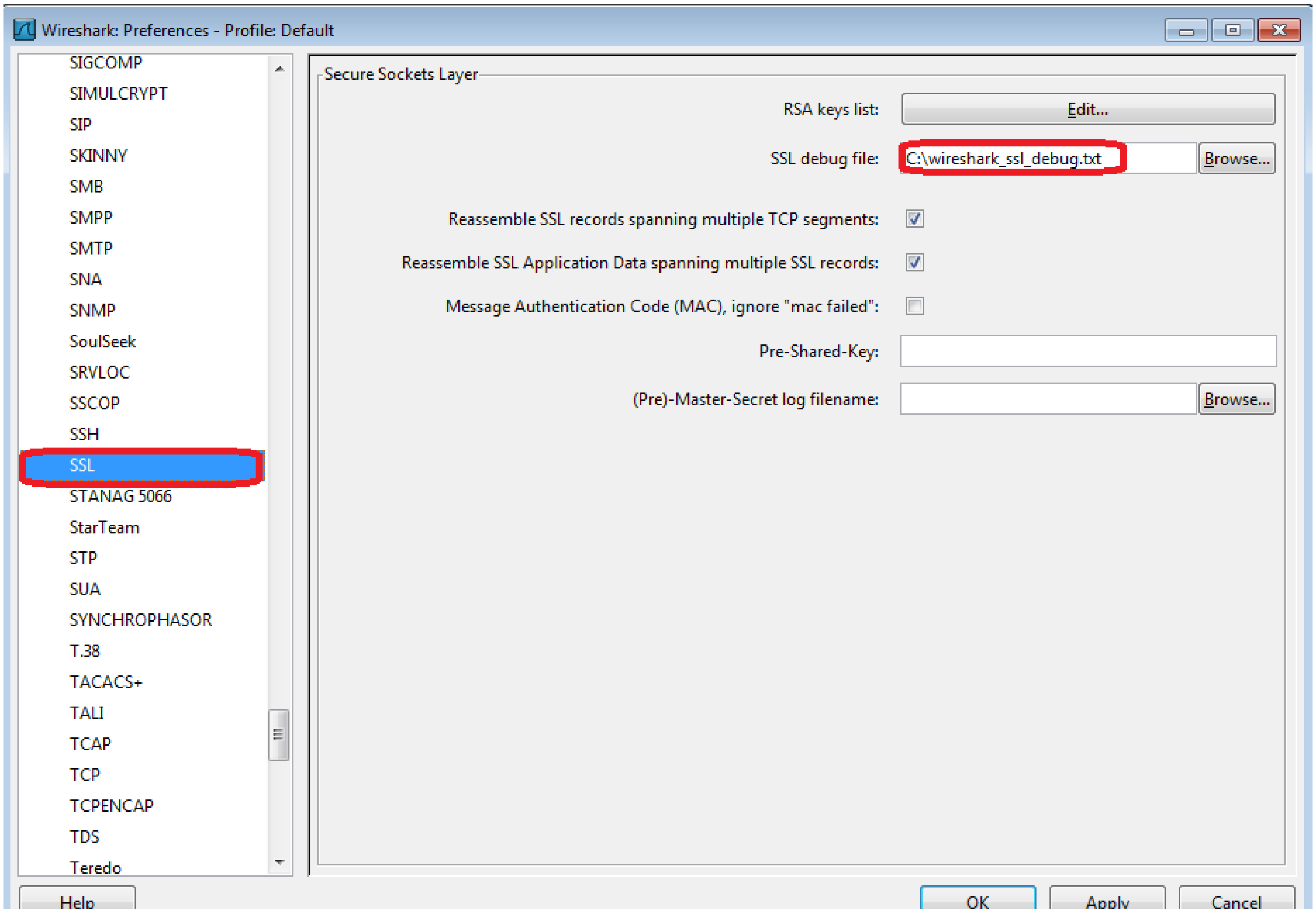
And press OK

3. Load the PEM key (see 2nd bullet of Input data section) to Wireshark: Edit => Preferences, find Protocols item in the left pane and lookup SSL



Display hidden protocol items: □

Make sure that SSL debug file is not empty, though the file name can be arbitrary, the path to file must exist. Press Edit button:



Secure Sockets Layer

RSA keys list:

SSL debug file: C:\wireshark_ssl_debug.txt

Reassemble SSL records spanning multiple TCP segments: ☑

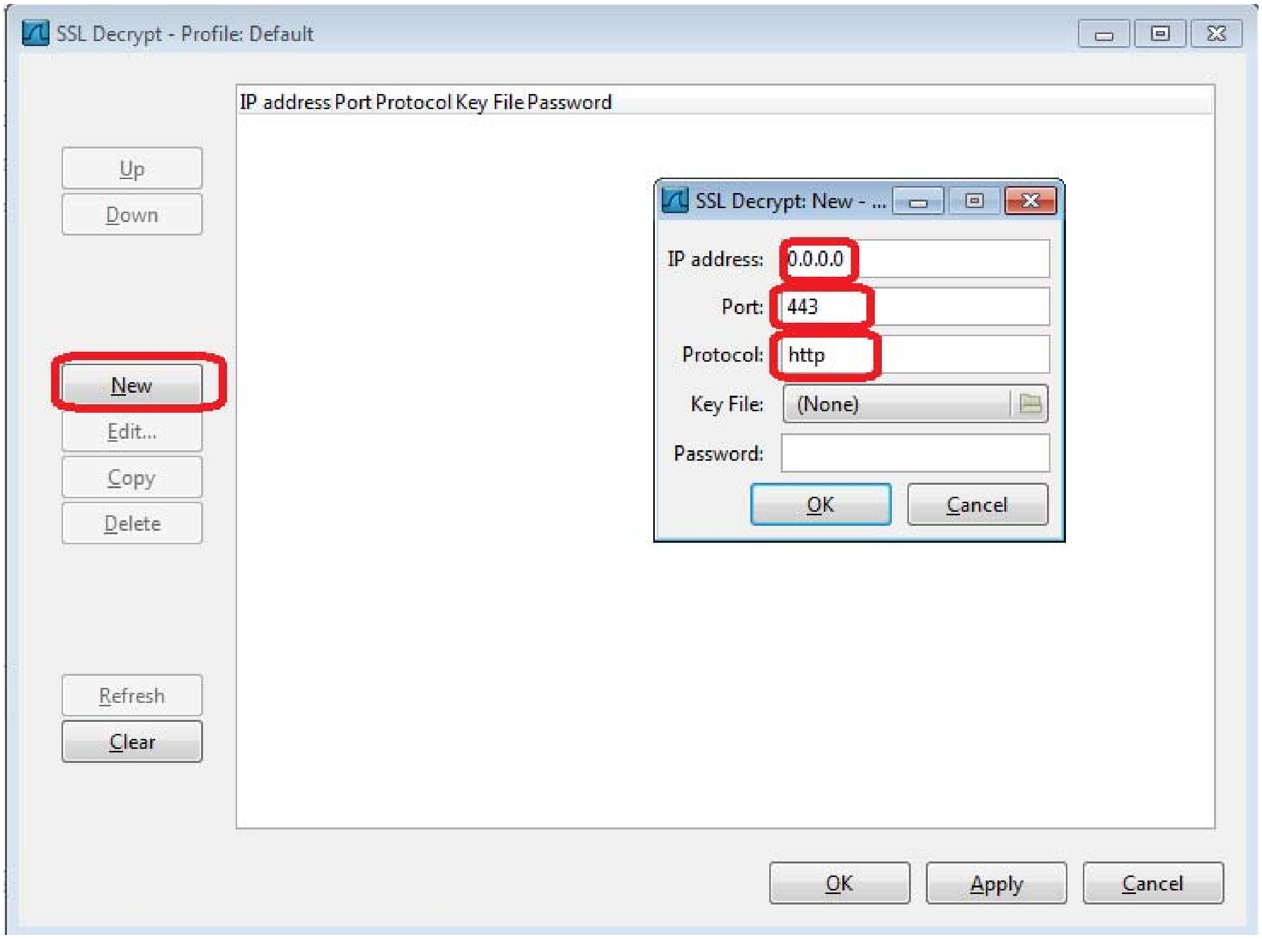
Reassemble SSL Application Data spanning multiple SSL records: ☑

Message Authentication Code (MAC), ignore "mac failed": □

Pre-Shared-Key:

(Pre)-Master-Secret log filename:

Press ‘New’ to add the key and fill in the field as it is shown below: IP address should be ‘0.0.0.0’, Port should match the application port (default is 443, but it could be different; it just should match the web/application server port for monitored application), let protocol be ‘http’ (lowercased). And push Key File button to select the PEM key [outputKey.pem] file. Leave Password empty, as we use unprotected PEM.



IP address: 0.0.0.0

Port: 443

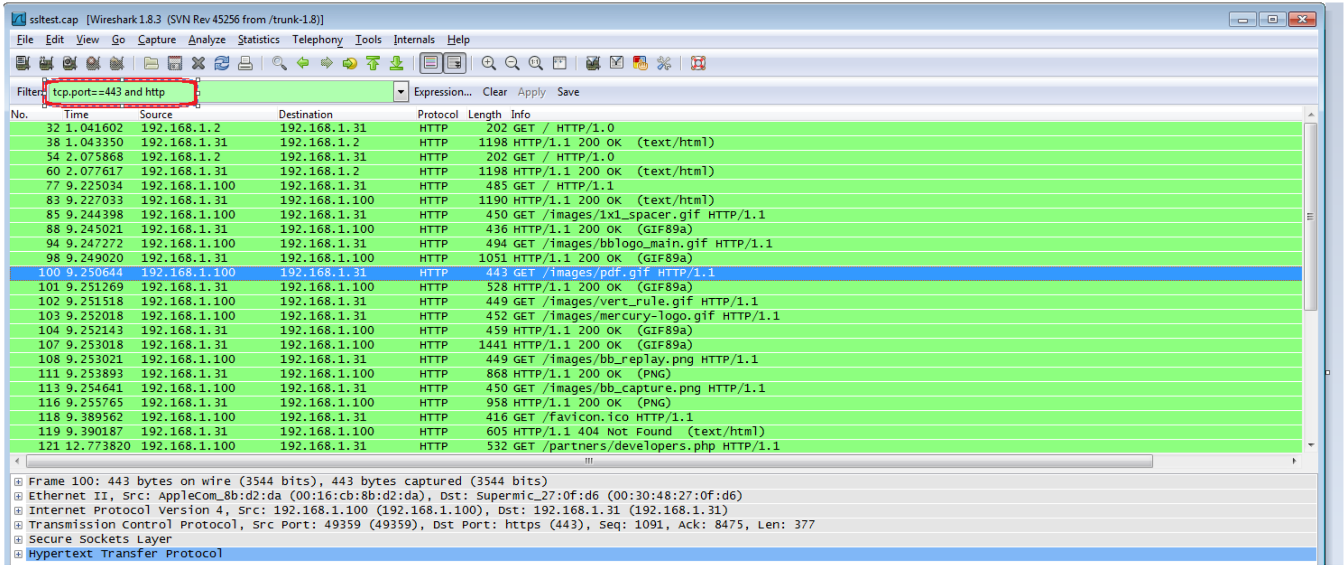
Protocol: http

Key File: (None)

Password:

Save the changes.

4. Apply ***tcp.port==443 and http*** filter and see if you can see some green packets appear (as it was mentioned above, use proper port number):


ssltest.cap [Wireshark 1.8.3 (SVN Rev 45256 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: tcp.port==443 and http

```
No.     Time      Source          Destination     Protocol Length Info
     32 1.041602  192.168.1.2     192.168.1.31    HTTP      202 GET / HTTP/1.0
     38 1.043350  192.168.1.31    192.168.1.2     HTTP     1198 HTTP/1.1 200 OK  (text/html)
     54 2.075868  192.168.1.2     192.168.1.31    HTTP      202 GET / HTTP/1.0
     60 2.077617  192.168.1.31    192.168.1.2     HTTP     1198 HTTP/1.1 200 OK  (text/html)
     77 9.225034  192.168.1.100   192.168.1.31    HTTP      485 GET / HTTP/1.1
     83 9.227033  192.168.1.31    192.168.1.100   HTTP     1190 HTTP/1.1 200 OK  (text/html)
     85 9.244398  192.168.1.100   192.168.1.31    HTTP      450 GET /images/1x1_spacer.gif HTTP/1.1
     88 9.245021  192.168.1.31    192.168.1.100   HTTP      436 HTTP/1.1 200 OK  (GIF89a)
     94 9.247272  192.168.1.100   192.168.1.31    HTTP      494 GET /images/bblogo_main.gif HTTP/1.1
     98 9.249020  192.168.1.31    192.168.1.100   HTTP     1051 HTTP/1.1 200 OK  (GIF89a)
    100 9.250644  192.168.1.100   192.168.1.31    HTTP      443 GET /images/pdf.gif HTTP/1.1
    101 9.251269  192.168.1.31    192.168.1.100   HTTP      528 HTTP/1.1 200 OK  (GIF89a)
    102 9.251518  192.168.1.100   192.168.1.31    HTTP      449 GET /images/vert_rule.gif HTTP/1.1
    103 9.252018  192.168.1.100   192.168.1.31    HTTP      452 GET /images/mercury-logo.gif HTTP/1.1
    104 9.252143  192.168.1.31    192.168.1.100   HTTP      459 HTTP/1.1 200 OK  (GIF89a)
    107 9.253018  192.168.1.31    192.168.1.100   HTTP     1441 HTTP/1.1 200 OK  (GIF89a)
    108 9.253021  192.168.1.100   192.168.1.31    HTTP      449 GET /images/bb_replay.png HTTP/1.1
    111 9.253893  192.168.1.31    192.168.1.100   HTTP      868 HTTP/1.1 200 OK  (PNG)
    113 9.254641  192.168.1.100   192.168.1.31    HTTP      450 GET /images/bb_capture.png HTTP/1.1
    116 9.255765  192.168.1.31    192.168.1.100   HTTP      958 HTTP/1.1 200 OK  (PNG)
    118 9.389562  192.168.1.100   192.168.1.31    HTTP      416 GET /favicon.ico HTTP/1.1
    119 9.390187  192.168.1.31    192.168.1.100   HTTP      605 HTTP/1.1 404 Not Found  (text/html)
    121 12.773820 192.168.1.100   192.168.1.31    HTTP      532 GET /partners/developers.php HTTP/1.1
```

```
⊞ Frame 100: 443 bytes on wire (3544 bits), 443 bytes captured (3544 bits)
⊞ Ethernet II, Src: AppleCom_8b:d2:da (00:16:cb:8b:d2:da), Dst: Supermic_27:0f:d6 (00:30:48:27:0f:d6)
⊞ Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 192.168.1.31 (192.168.1.31)
⊞ Transmission Control Protocol, Src Port: 49359 (49359), Dst Port: https (443), Seq: 1091, Ack: 8475, Len: 377
⊞ Secure Sockets Layer
⊞ Hypertext Transfer Protocol
```

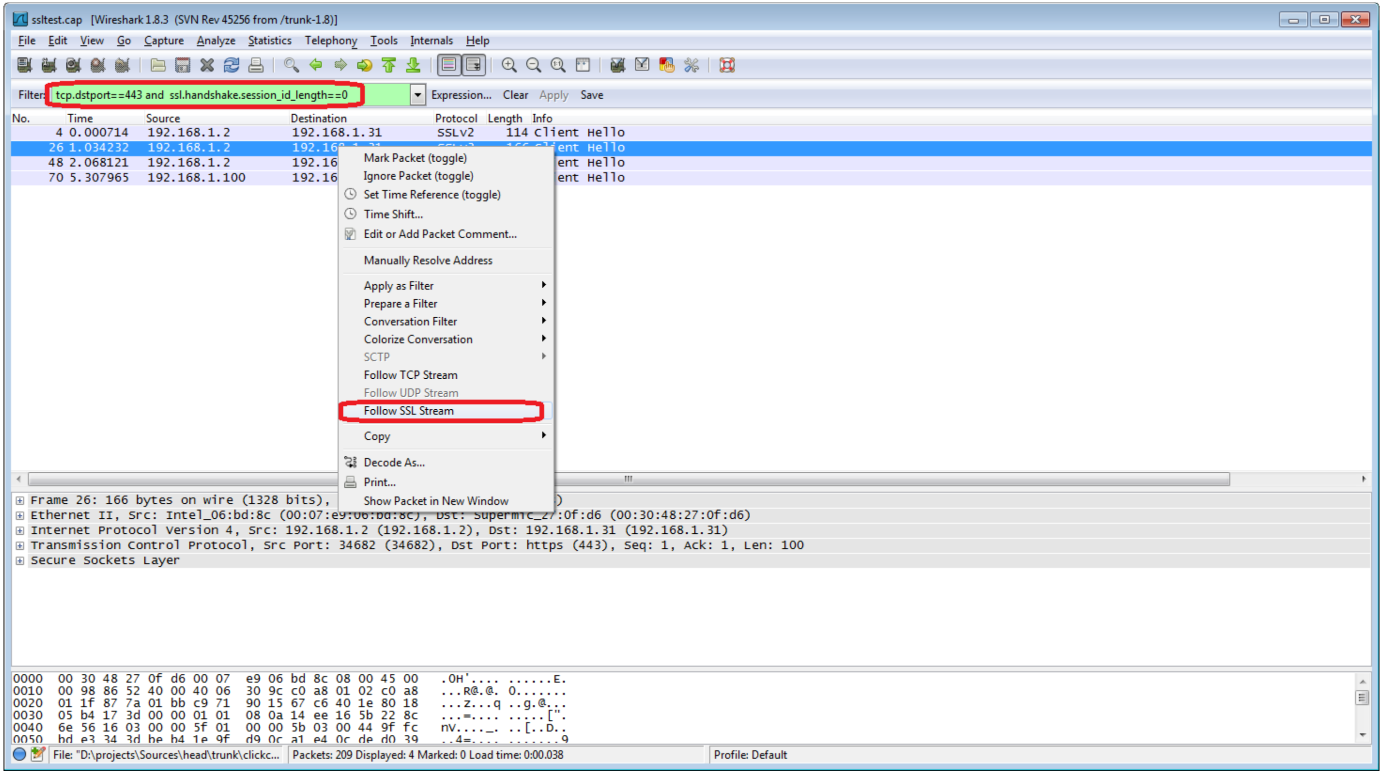
You can select a packet, right-click 'Follow SSL stream' and view decrypted content

Win!

If you do not see filtered packets, it means:

a. The decrypted protocol appeared not HTTP. Try next steps to discovered whether SSL decryption was successful:
    a. Apply ***tcp.dstport==443 and ssl.handshake.session_id_length==0*** filter (again, please take care about proper port number)
    b. If the filtered packet list is empty it means that the captured pcap file contains no full SSL handshake, so the traffic could not be decrypted. Ask the customer to record another capture file.
    c. If the list is not empty, for some of the packets perform next:
        i. Right-click and select 'Follow SSL stream' (please do not confuse with 'Follow TCP stream')
        ii. If Stream Content is not empty, it's Win, the traffic was decrypted successfully!
        iii. If not, repeat a.c.i. for next packet (need to re-apply the filter a.a. again)

If you did not manage to get not empty Stream Content for several filtered packets, it's fail, proceed to next step 4.b.



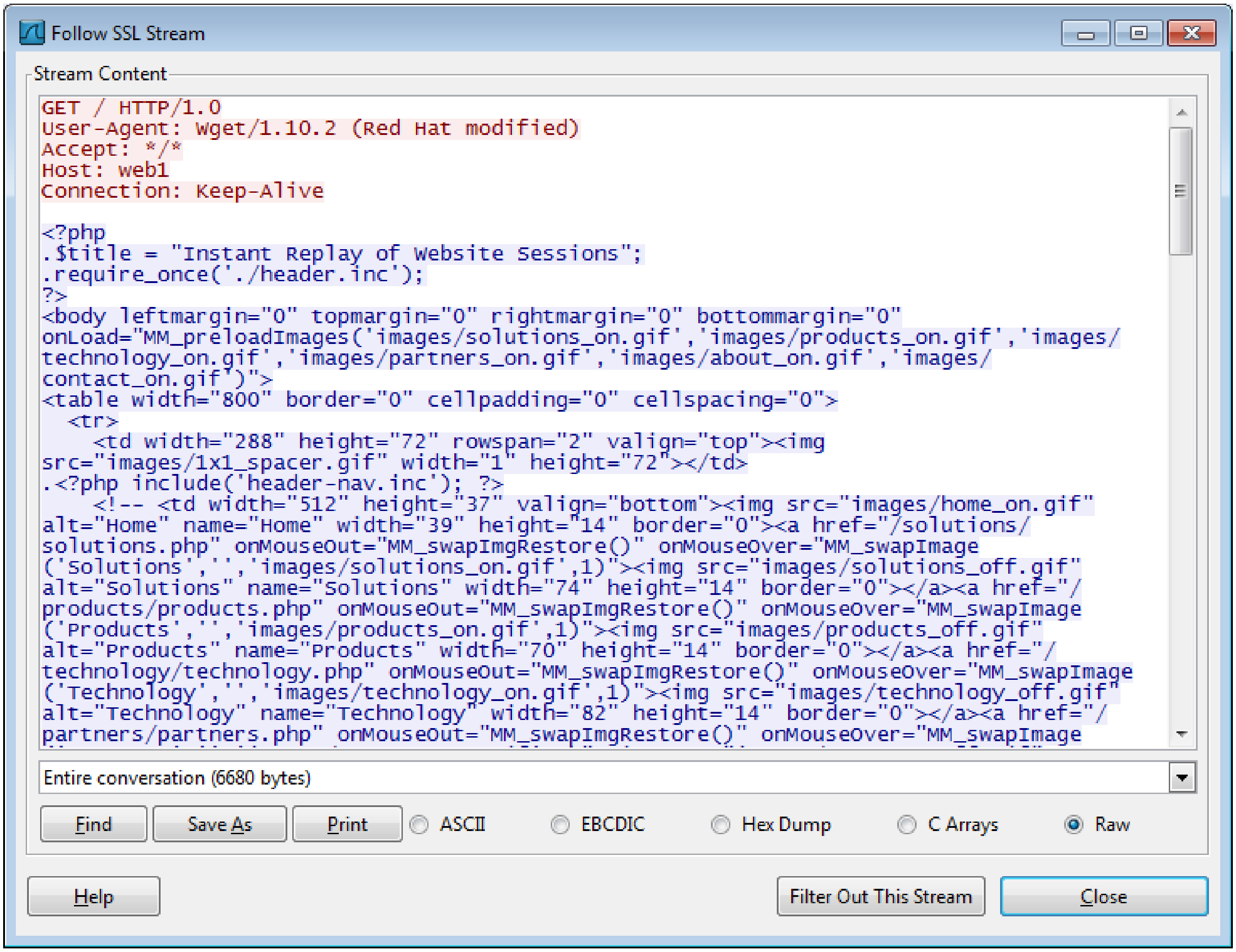
Filter: tcp.dstport==443 and ssl.handshake.session_id_length==0

```
No.     Time        Source           Destination      Protocol  Length  Info
      4 0.000714    192.168.1.2      192.168.1.31     SSLv2        114  Client Hello
     26 1.034232    192.168.1.2      192.16                                ent Hello
     48 2.068121    192.168.1.2      192.16                                ent Hello
     70 5.307965    192.168.1.100    192.16                                ent Hello
```

```
⊞ Frame 26: 166 bytes on wire (1328 bits),                                   )
⊞ Ethernet II, Src: Intel_06:bd:8c (00:07:e9:06:bd:8c), Dst: Supermic_27:0f:d6 (00:30:48:27:0f:d6)
⊞ Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.31 (192.168.1.31)
⊞ Transmission Control Protocol, Src Port: 34682 (34682), Dst Port: https (443), Seq: 1, Ack: 1, Len: 100
⊞ Secure Sockets Layer
```

```
0000  00 30 48 27 0f d6 00 07  e9 06 bd 8c 08 00 45 00   .0H'.... ......E.
0010  00 98 86 52 40 00 40 06  30 9c c0 a8 01 02 c0 a8   ...R@.@. 0.......
0020  01 1f 87 7a 01 bb c9 71  90 15 67 c6 40 1e 80 18   ...z...q ..g.@...
0030  05 b4 17 3d 00 00 01 01  08 0a 14 ee 16 5b 22 8c   ...=.... .....[".
0040  6e 56 16 03 00 00 5f 01  00 00 5b 03 00 44 9f fc   nV...._. ..[..D..
0050  bd e3 34 3d be b4 1e 9f  d9 0c a1 e4 0c de d0 39   ..4=.... .......9
```

File: "D:\projects\Sources\head\trunk\clickc... Packets: 209 Displayed: 4 Marked: 0 Load time: 0:00.038 Profile: Default



```
GET / HTTP/1.0
User-Agent: Wget/1.10.2 (Red Hat modified)
Accept: */*
Host: web1
Connection: Keep-Alive

<?php
.$title = "Instant Replay of Website Sessions";
.require_once('./header.inc');
?>
<body leftmargin="0" topmargin="0" rightmargin="0" bottommargin="0"
onLoad="MM_preloadImages('images/solutions_on.gif','images/products_on.gif','images/
technology_on.gif','images/partners_on.gif','images/about_on.gif','images/
contact_on.gif')">
<table width="800" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="288" height="72" rowspan="2" valign="top"><img
src="images/1x1_spacer.gif" width="1" height="72"></td>
.<?php include('header-nav.inc'); ?>
    <!-- <td width="512" height="37" valign="bottom"><img src="images/home_on.gif"
alt="Home" name="Home" width="39" height="14" border="0"><a href="/solutions/
solutions.php" onMouseOut="MM_swapImgRestore()" onMouseOver="MM_swapImage
('Solutions','','images/solutions_on.gif',1)"><img src="images/solutions_off.gif"
alt="Solutions" name="Solutions" width="74" height="14" border="0"></a><a href="/
products/products.php" onMouseOut="MM_swapImgRestore()" onMouseOver="MM_swapImage
('Products','','images/products_on.gif',1)"><img src="images/products_off.gif"
alt="Products" name="Products" width="70" height="14" border="0"></a><a href="/
technology/technology.php" onMouseOut="MM_swapImgRestore()" onMouseOver="MM_swapImage
('Technology','','images/technology_on.gif',1)"><img src="images/technology_off.gif"
alt="Technology" name="Technology" width="82" height="14" border="0"></a><a href="/
partners/partners.php" onMouseOut="MM_swapImgRestore()" onMouseOver="MM_swapImage
```

Entire conversation (6680 bytes)

○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

b. The Wireshark failed to decrypt the traffic, most probably because of incorrect private key or unsupported SSL algorithm. In this case ask CPE engineers for help.

Please find attached pcap and private key to try the steps. The private key is already in unprotected PEM format, so no need to convert it.

ssl_test.zip