

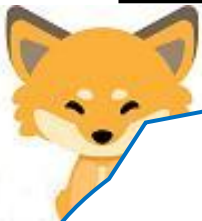


```

└──(root@kali)-[/home/kali]
└─# ifconfig eth0 promisc ( Enter promiscuous mode.)

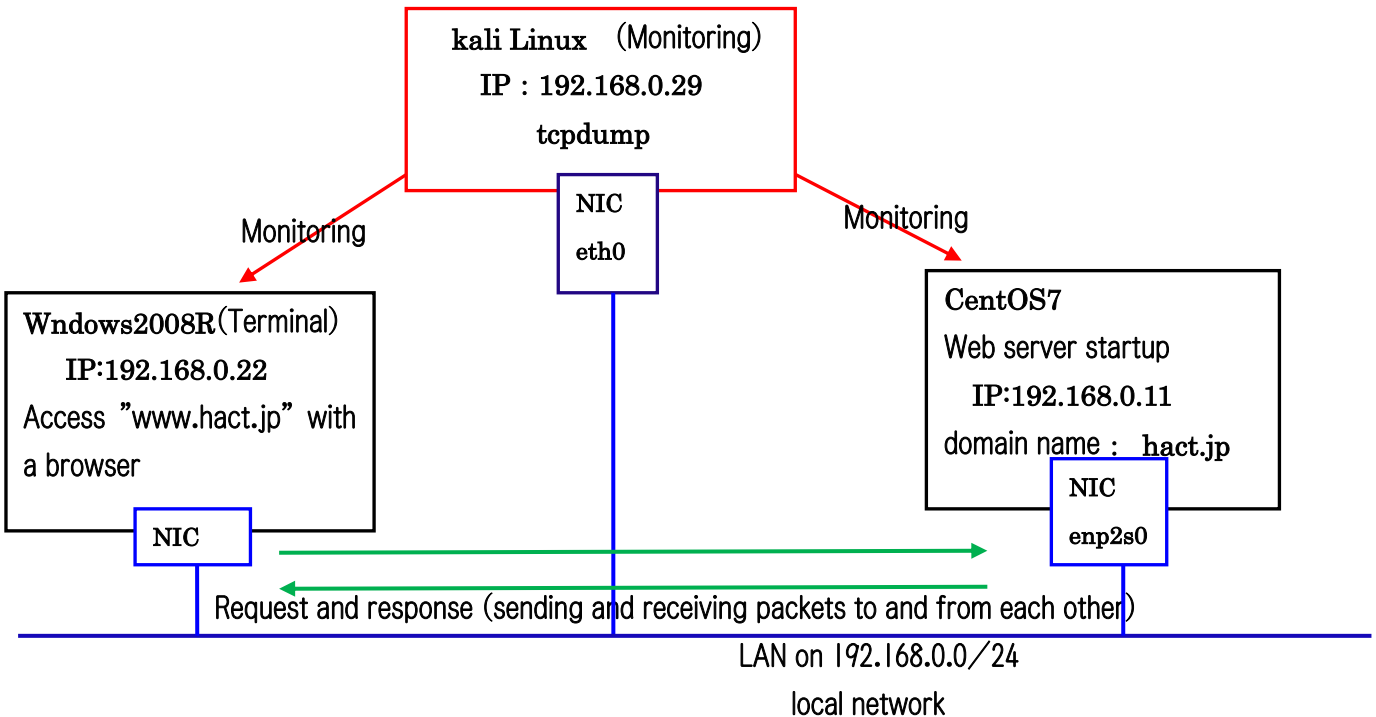
└──(root@kali)-[/home/kali]
└─# ifconfig eth0 ( Confirmation that you are in promiscuous mode )
eth0: flags=4419<UP, BROADCAST, RUNNING, PROMISC, MULTICAST> mtu 1500
    inet 192.168.0.29 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::20c:29ff:fe7f:57d prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:7f:05:7d txqueuelen 1000 (Ethernet)
    RX packets 2482 bytes 651699 (636.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 202 bytes 16020 (15.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```



Next, "kali Linux" monitors the status of packet communication between the CentOS7 web server and the Windows 2008R browser using the tcpdump tool. The following figure is a simple illustration.

Before using the tcpdump tool, set the NIC (device name: eth0) to promiscuous mode (non-discriminatory mode), which captures all packets coming into the NIC. It operates as follows.





The results (in part) monitored by the Internet "kali Linux" with tcpdump are shown below. What is important here is the degree to which the contents of the packets being sent and received can be read.

### [Web Server]

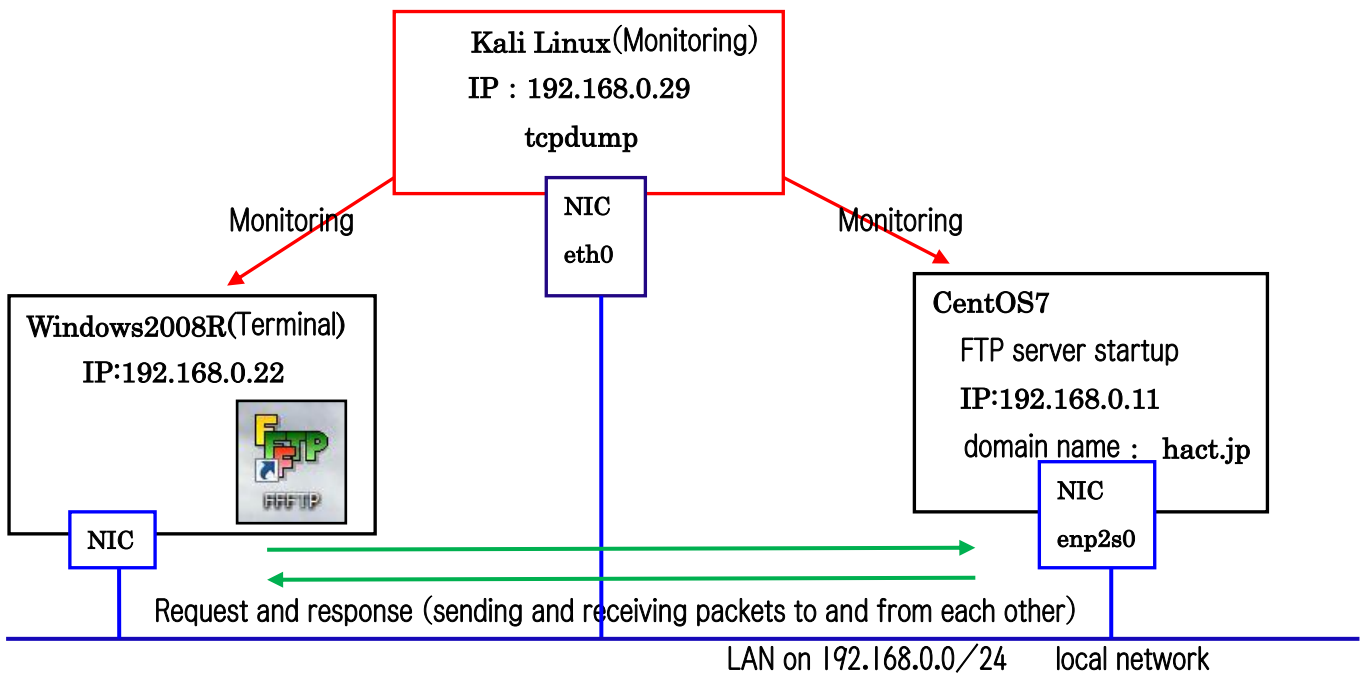
```
(root@kali)-[/home/kali]
└─# tcpdump -l -X port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:35:08.559231 IP 192.168.0.22.1083 > ns.hact.jp.http: Flags [S], seq 422337074, win 8192,
options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
    0x0000:  4500 0034 03fa 4000 8006 7558 c0a8 0016  E..4..@...uX...
    0x0010:  c0a8 000b 043b 0050 192c 5a32 0000 0000  .....:..P.,Z2...
    0x0020:  8002 2000 55b5 0000 0204 05b4 0103 0308  ....U.....
    0x0030:  0101 0402                                     ....
        :
        :      Omission
        :
17:35:08.559234 IP 192.168.0.22.1083 > ns.hact.jp.http: Flags [P.], seq 1:448, ack 1, win 256,
length 447: HTTP: GET /hact.html HTTP/1.1
    0x0000:  4500 01e7 03fc 4000 8006 73a3 c0a8 0016  E.....@...s.....
    0x0010:  c0a8 000b 043b 0050 192c 5a33 8988 bc57  .....:..P.,Z3...W
    0x0020:  5018 0100 a8ea 0000 4745 5420 2f68 6163  P.....GET./hac
    0x0030:  742e 6874 6d6c 2048 5454 502f 312e 310d  t.html.HTTP/1.1.
    0x0040:  0a48 6f73 743a 2077 7777 2e68 6163 742e  .Host:..www.hact.
    0x0050:  6a70 0d0a 5573 6572 2d41 6765 6e74 3a20  jp..User-Agent:.
    0x0060:  4d6f 7a69 6c6c 612f 352e 3020 2857 696e  Mozilla/5.0.(Win
    0x0070:  646f 7773 204e 5420 362e 313b 2057 696e  dows.NT.6.1;.Win
    0x0080:  3634 3b20 7836 343b 2072 763a 3130 302e  64;.x64;.rv:100.
    0x0090:  3029 2047 6563 6b6f 2f32 3031 3030 3130  0).Gecko/2010010
    0x00a0:  3120 4669 7265 666f 782f 3130 302e 300d  1.Firefox/100.0.
    0x00b0:  0a41 6363 6570 743a 2074 6578 742f 6874  .Accept:..text/ht
    0x00c0:  6d6c 2c61 7070 6c69 6361 7469 6f6e 2f78  ml,application/x
    0x00d0:  6874 6d6c 2b78 6d6c 2c61 7070 6c69 6361  html+xml, applica
        :
        :      Omission
```



t reads! This is a request for the hact.html file from Windows 2008R (IP: 192.168.0.22) to the CentOS7 web server (ns.hact.jp) via http. I see that the red "hact.html" corresponds to the hexadecimal ASCII code "686163742e68746d6c". So we need the first ASCII code table. Surely, it's a lot of work to find out if there is any important information hidden from this ASCII code. With https, even the requested hact.html file is not displayed, so it is safe.



Next is FTP. The communication status is illustrated below. The application used on the Windows 2008R side is FFFTP. Since there is no encryption or anything, it is fully visible!





The results of the monitoring were as follows.

[FTP]

```

└──(root@kali)-[/home/kali]
└─# tcpdump -l -X 'ip host 192.168.0.11'
18:01:14.316507 IP 192.168.0.22.1088 > ns.hact.jp.ftp: Flags [P.], seq 1:12, ack 21, win 256, length
11: FTP: USER hact ← User name : hact
0x0000: 4500 0033 0167 4000 8006 77ec c0a8 0016 E..3.g@...w....
0x0010: c0a8 000b 0440 0015 2abe 4339 a721 04e1 .....@..*.C9.!..
0x0020: 5018 0100 7482 0000 5553 4552 2068 6163 P...t...USER.hac
0x0030: 740d 0a                                     t..
18:01:14.316507 IP ns.hact.jp.ftp > 192.168.0.22.1088: Flags [.], ack 12, win 229, length 0
0x0000: 4500 0028 95be 4000 4006 23a0 c0a8 000b E..(.@.@.#....
0x0010: c0a8 0016 0015 0440 a721 04e1 2abe 4344 .....@.!..*.CD
0x0020: 5010 00e5 0f24 0000 0000 0000 0000 P....$.
18:01:14.316592 IP ns.hact.jp.ftp > 192.168.0.22.1088: Flags [P.], seq 21:55, ack 12, win 229, length
34: FTP: 331 Please specify the password.
0x0000: 4500 004a 95bf 4000 4006 237d c0a8 000b E..J..@.@.#}....
0x0010: c0a8 0016 0015 0440 a721 04e1 2abe 4344 .....@.!..*.CD
0x0020: 5018 00e5 c7d8 0000 3333 3120 506c 6561 P.....331.Plea
0x0030: 7365 2073 7065 6369 6679 2074 6865 2070 se.specify.the.p
0x0040: 6173 7377 6f72 642e 0d0a                                     assword...
18:01:14.316848 IP 192.168.0.22.1088 > ns.hact.jp.ftp: Flags [P.], seq 12:26, ack 55, win 256, length
14: FTP: PASS 0515Hac
0x0000: 4500 0036 0168 4000 8006 77e8 c0a8 0016 E..6.h@...w....
0x0010: c0a8 000b 0440 0015 2abe 4344 a721 0503 .....@..*.CD.!..
0x0020: 5018 0100 7225 0000 5041 5353 2030 3531 P...r%..PASS.051 ← password
0x0030: 3548 6163 0d0a                                     5Hac..
                                : Omission

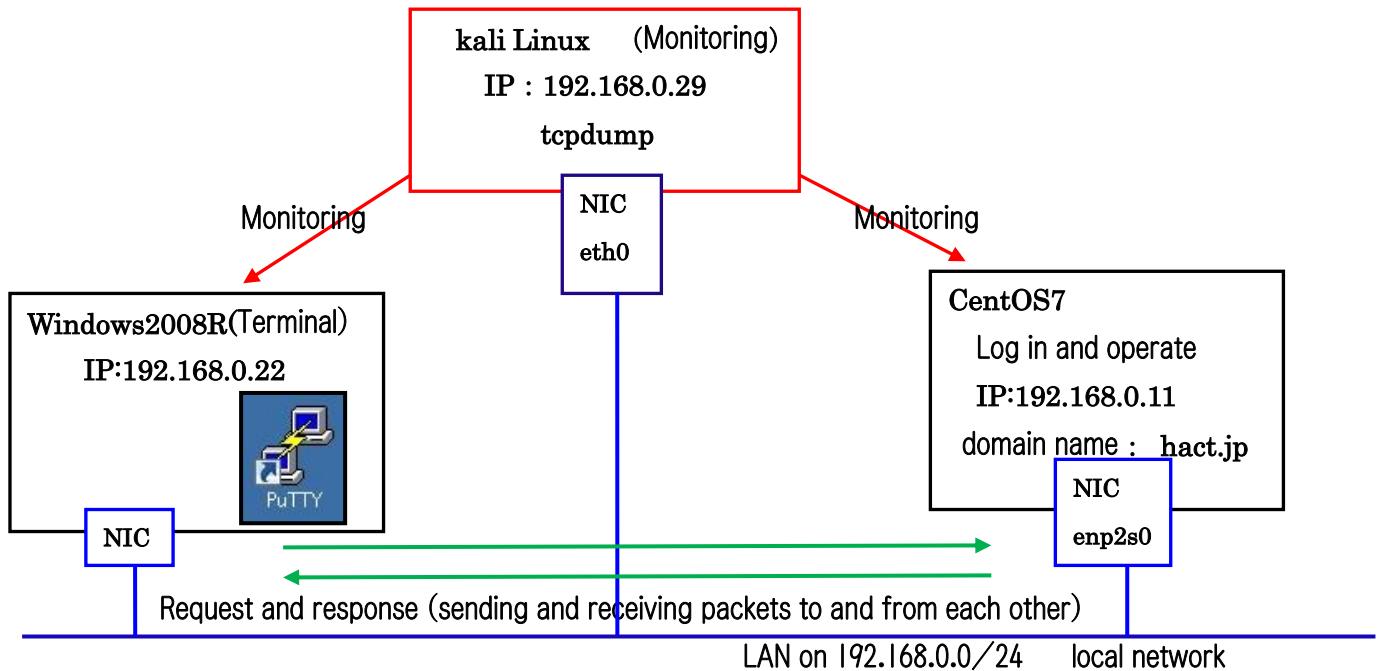
```



I see, the user name and password are completely visible!  
 If this user has administrative privileges (which I usually give them),  
 they can completely hijack the server. It's not good!



The next step is remote control from a terminal (Windows 2008R). The following shows the difference between Telnet and SSH connections from the terminal to CentOS7. As you might expect, Telnet is visible, but SSH should be almost invisible. The application used is PuTTY.



Initially [Telnet].

```
(root@kali)-[~/kali]
└─# tcpdump -l -X port 23
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:15:58.731262 IP 192.168.0.22.socks > ns.hact.jp.telnet: Flags [S], seq 3493127081, win 8192,
: Omission
18:16:28.423996 IP ns.hact.jp.telnet > 192.168.0.22.socks: Flags [P.], seq 125:135, ack 83, win 229,
length 10
    0x0000: 4510 0032 a666 4000 4006 12de c0a8 000b  E..2.f@.@.....
    0x0010: c0a8 0016 0017 0438 ccf9 89d6 d034 e3fc  ....8....4..
    0x0020: 5018 00e5 3652 0000 5061 7373 776f 7264  P...6R..Password
    0x0030: 3a20                                     :.
18:16:28.623458 IP 192.168.0.22.socks > ns.hact.jp.telnet: Flags [.], ack 135, win 256, length 0
    0x0000: 4500 0028 0224 4000 8006 773a c0a8 0016  E..(.$@...w:...
    0x0010: c0a8 000b 0438 0017 d034 e3fc ccf9 89e0  ....8...4.....
    0x0020: 5010 0100 1e08 0000 0000 0000 0000 0000  P.....
: Omission
```



Unlike ftp, with Telenet, the message from CentOS7 requesting Password:. (506l 7373 776f 7264 3a20) request message is displayed, but the password (05l5Hac) sent from the terminal (PuTTY) is once passed to the socket, encrypted (4500 0028 0224 40) and invisible, right? I can't find the username being sent either, so I guess that's a relief!



The next state is when connecting via SSH from PuTTY.

Next [SSH].

```

└──(root@kali)-[/home/kali]
└─# tcpdump -l -X port 22
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:24:07.316583 IP 192.168.0.22.1083 > ns.hact.jp.ssh: Flags [S], seq 1460512539, win 8192,
options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
    0x0000: 4500 0034 02b5 4000 8006 769d c0a8 0016  E..4..@...v.....
    0x0010: c0a8 000b 043b 0016 570d a71b 0000 0000  ....:..W.....
    0x0020: 8002 2000 cb24 0000 0204 05b4 0103 0308  ....$.
    0x0030: 0101 0402
                                     : Omission
18:24:07.436605 IP ns.hact.jp.ssh > 192.168.0.22.1083: Flags [P.], seq 1510:1574, ack 1325, win
251, length 64
    0x0000: 4500 0068 8c5f 4000 4006 2cbf c0a8 000b  E..h._@.@.....
    0x0010: c0a8 0016 0016 043b ad53 2ec6 570d ac48  ....:..S..W..H
    0x0020: 5018 00fb c10f 0000 2127 c44e 52ca 83d6  P.....!'..NR...
    0x0030: 884d 4cb7 c8c8 c5ca b3da 31f4 6ddc 0d46  .ML.....1.m..F
    0x0040: 6aed f9d0 2589 21cf 0a98 0d48 9664 f280  j...%!....H.d..
    0x0050: 51f5 4c6e ee17 f51e 15e9 adca db6e 430d  Q.Ln.....nC.
    0x0060: 2845 1ec7 fe77 1184
                                     (E...w..
18:24:07.640193 IP 192.168.0.22.1083 > ns.hact.jp.ssh: Flags [.], ack 1574, win 256, length 0
    0x0000: 4500 0028 02bb 4000 8006 76a3 c0a8 0016  E..(.@...v.....
    0x0010: c0a8 000b 043b 0016 570d ac48 ad53 2f06  ....:..W..H.S/.
    0x0020: 5010 0100 4962 0000 0000 0000 0000
    P...Ib.....
                                     : Omission

```

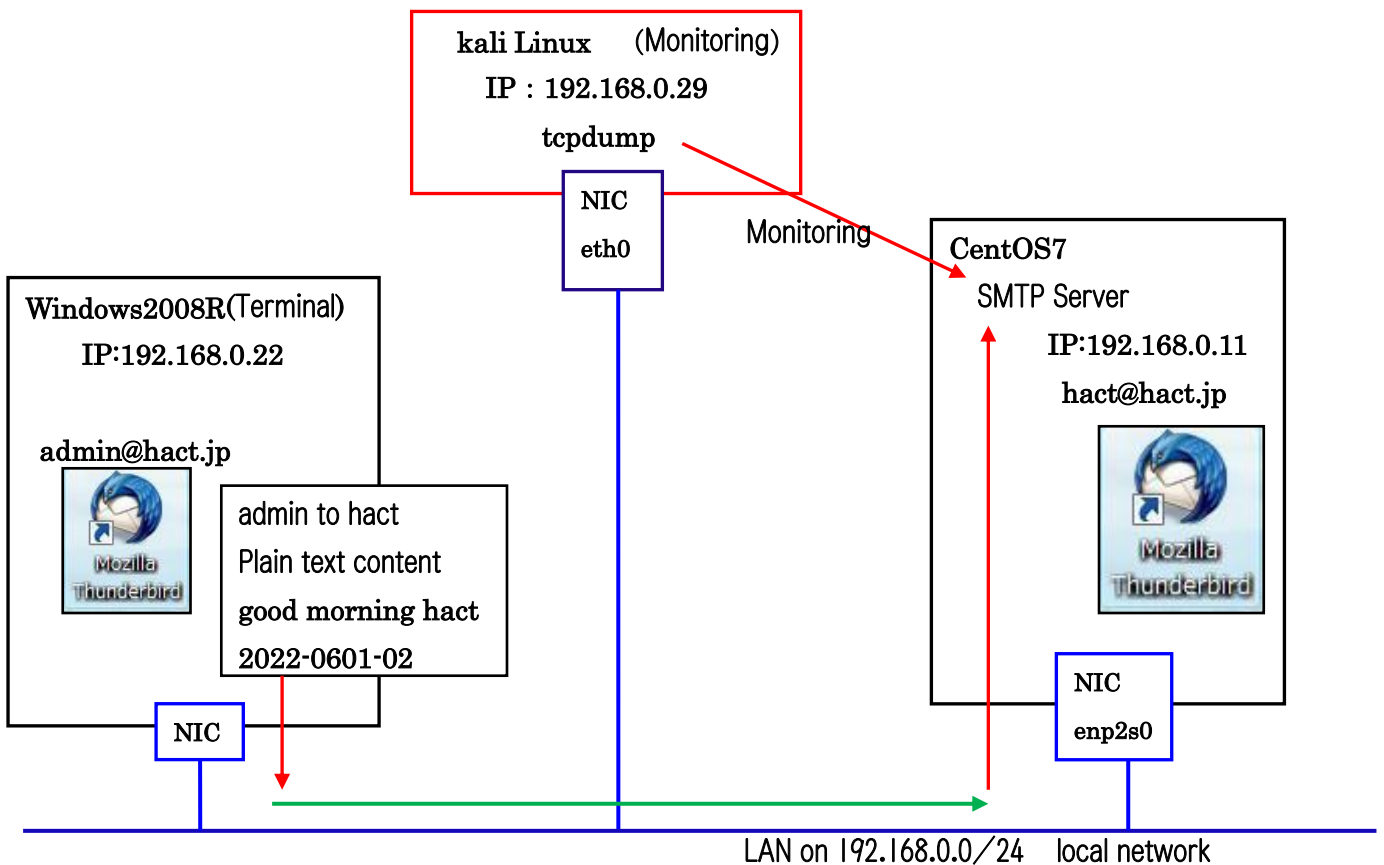


I see. In the case of SSH, the password request message itself is encrypted and invisible, so the location of the password cannot be identified.

In this case, we know the location of the password because we are comparing it with Telnet, but it is usually impossible to identify the password. Naturally, the password is also encrypted. This is a safe bet.



The next step is to send and receive email. The next step is to show how dangerous it is to send and receive e-mail messages in unencrypted plain text. The first step is to monitor the SMTP server, which is a server for sending e-mails (port number: 25), so "kali Linux" captures the moment of transmission. The following figure shows the sending and receiving of the message.







The captured results were as follows.

First, the outgoing server [SMTP].

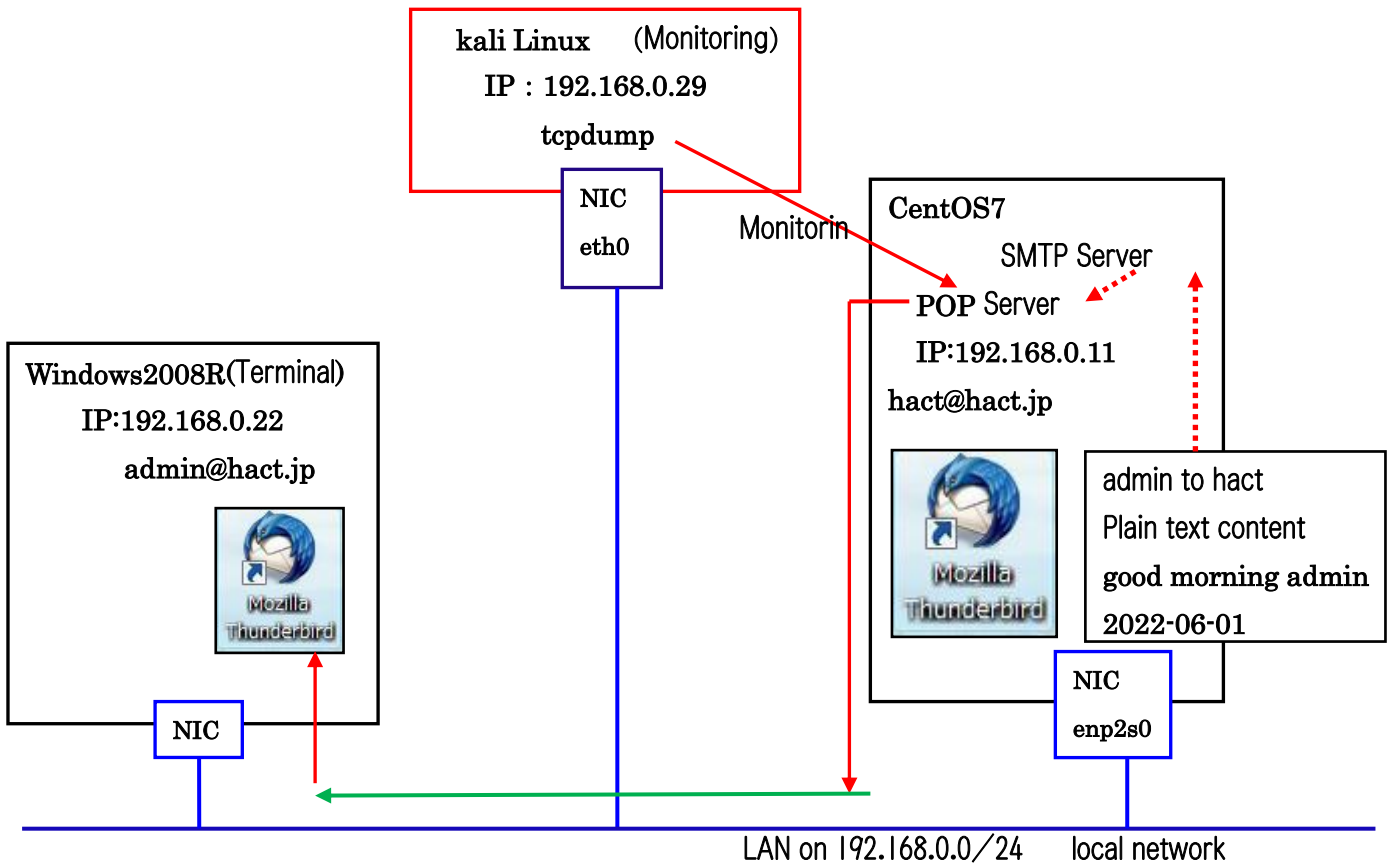
```
(root@kali)~# tcpdump -l -X port 25
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:38:37.950974 IP 192.168.0.22.1190 > ns.hact.jp.smtp: Flags [S], seq 1842926453, win 8192,
options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
 0x0000: 4500 0034 042a 4000 8006 7528 c0a8 0016  E..4.*@...u(....
 0x0010: c0a8 000b 04a6 0019 6dd8 d375 0000 0000  ....m..u....
 0x0020: 8002 2000 8791 0000 0204 05b4 0103 0308  ....
 0x0030: 0101 0402
                                     : Omission
 0x0020: 5018 0100 bae1 0000 4d41 494c 2046 524f  P.....MAIL.FRO
 0x0030: 4d3a 3c61 646d 696e 4068 6163 742e 6a70  M:<admin@hact.jp
 0x0040: 3e20 424f 4459 3d38 4249 544d 494d 4520  >.BODY=8BITMIME.
 0x0050: 5349 5a45 3d34 3333 0d0a                SIZE=433..
                                     : Omission
 0x0020: 5018 0100 520b 0000 5243 5054 2054 4f3a  P...R...RCPT.TO:
 0x0030: 3c68 6163 7440 6861 6374 2e6a 703e 0d0a  <hact@hact.jp>..
                                     : Omission
 0x0130: 4d45 2d56 6572 7369 6f6e 3a20 312e 300d  ME-Version:.1.0.
 0x0140: 0a43 6f6e 7465 6e74 2d54 7970 653a 2074  .Content-Type:.t
 0x0150: 6578 742f 706c 6169 6e3b 2063 6861 7273  ext/plain;.chars
 0x0160: 6574 3d75 7466 2d38 3b20 666f 726d 6174  et=utf-8;.format
 0x0170: 3d66 6c6f 7765 640d 0a43 6f6e 7465 6e74  =flowed..Content
 0x0180: 2d54 7261 6e73 6665 722d 456e 636f 6469  -Transfer-Encodi
 0x0190: 6e67 3a20 3762 6974 0d0a 436f 6e74 656e  ng:.7bit..Conten
 0x01a0: 742d 4c61 6e67 7561 6765 3a20 656e 2d55  t-Language:.en-U
 0x01b0: 530d 0a0d 0a67 6f6f 6420 6d6f 726e 696e  S....good.mornin
 0x01c0: 6720 6861 6374 2032 3032 322d 3036 3031  g.hact.2022-0601
 0x01d0: 2d30 320d 0a0d 0a0d 0a2e 0d0a          -02.....
                                     : Omission
```



Kitsune, this is not good. The body of the message is visible, and both the sender's and recipient's email addresses are visible.



That's right, though. Let's look at the results of the scan when the incoming POP server (port number: 110) was monitored.



Next, the server for receiving [POP].

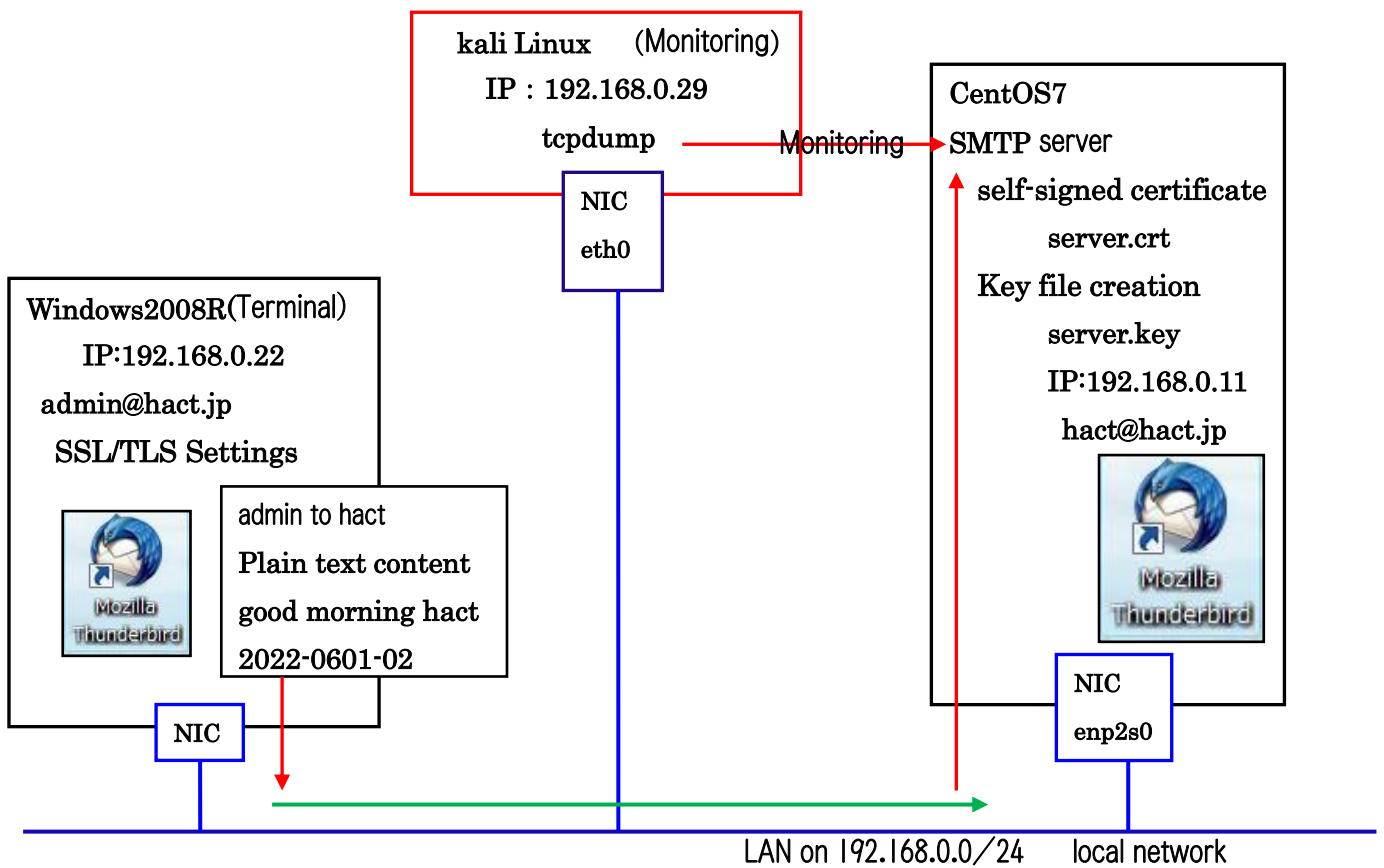
```
(root@kali)~# tcpdump -l -X port 110
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:28:34.833850 IP 192.168.0.22.1182 > ns.hact.jp.pop3: Flags [S], seq 2250811765, win
8192, options [mss 1460,nop,wscale 8,nop,nop,sack0K], length 0
    0x0000: 4500 0034 039d 4000 8006 75b5 c0a8 0016  E..4..@...u....
    0x0010: c0a8 000b 049e 006e 8628 a975 0000 0000  .....n.(u....
    0x0020: 8002 2000 98f4 0000 0204 05b4 0103 0308  .....
    0x0030: 0101 0402                                     ....
                                     : Omission
18:28:35.187847 IP ns.hact.jp.pop3 > 192.168.0.22.1182: Flags [P.], seq 245:945, ack 71,
win 229, length 70
                                     : Omission
    0x01f0: 3031 0d0a 2054 6875 6e64 6572 6269 7264  01...Thunderbird
    0x0200: 2f39 312e 392e 300d 0a43 6f6e 7465 6e74  /91.9.0..Content
    0x0210: 2d4c 616e 6775 6167 653a 2065 6e2d 5553  -Language:.en-US
    0x0220: 0d0a 546f 3a20 6164 6d69 6e40 6861 6374  ..To:.admin@hact
    0x0230: 2e6a 700d 0a46 726f 6d3a 2048 6163 5420  .jp..From:.HacT.
    0x0240: 3c68 6163 7440 6861 6374 2e6a 703e 0d0a  <hact@hact.jp>..
    0x0250: 5375 626a 6563 743a 2067 6f61 6973 6174  Subject:.goaisat
    0x0260: 7521 0d0a 436f 6e74 656e 742d 5479 7065  u!..Content-Type
    0x0270: 3a20 7465 7874 2f70 6c61 696e 3b20 6368  :.text/plain;.ch
    0x0280: 6172 7365 743d 5554 462d 383b 2066 6f72  arset=UTF-8;.for
    0x0290: 6d61 743d 666c 6f77 6564 0d0a 436f 6e74  mat=flowed..Cont
    0x02a0: 656e 742d 5472 616e 7366 6572 2d45 6e63  ent-Transfer-Enc
    0x02b0: 6f64 696e 673a 2037 6269 740d 0a0d 0a67  oding:.7bit....g
    0x02c0: 6f6f 6420 6d6f 726e 696e 6720 6164 6d69  ood.morning.admi
    0x02d0: 6e20 3230 3232 2d30 3630 310d 0a0d 0a0d  n.2022-0601.....
    0x02e0: 0a2e 0d0a                                     ....
                                     : Omission
```



Kitsune, the receiving end (POP) is clearly visible too. What about encryption?



Let's try encrypting with SSL/TLS!  
The monitoring process is the same as before encryption.  
First, SMTP (port number: 465).



## [SMTP] with SSL/TLS

```
└──(root@kali)-[/home/kali]
```

```
└─# tcpdump -l -X port 465
```

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
```

```
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

```
11:55:01.148918 IP 192.168.0.22.1081 > ns.hact.jp.submissions: Flags [S], seq 2067232394, win 8192,
```

```
options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
```

```
0x0000: 4500 0034 03c2 4000 8006 7590 c0a8 0016 E..4..@...u....
```

```
0x0010: c0a8 000b 0439 01d1 7b37 768a 0000 0000 .....9..{7v....
```

```
0x0020: 8002 2000 d5d2 0000 0204 05b4 0103 0308 .....
```

```
0x0030: 0101 0402
```

```
: Omission
```

```
11:55:01.248681 IP ns.hact.jp.submissions > 192.168.0.22.1081: Flags [P.], seq 1:1432, ack 518, win 237, length 1431
```

```
0x0000: 4500 05bf 7ed4 4000 4006 34f3 c0a8 000b E...~.@.@.4....
```

```
0x0010: c0a8 0016 01d1 0439 9b39 544a 7b37 7890 .....9.9TJ{7x.
```

```
0x0020: 5018 00ed 9044 0000 1603 0300 3d02 0000 P....D.....=...
```

```
0x0030: 3903 03b4 11b4 54ba fc2a 3a95 4a7c fc53 9....T..*:.J|.S
```

```
0x0040: 8ae7 23d4 108f 6b3b 742c 3ce5 2e67 e46d ..#...k;t,<..g.m
```

```
0x0050: f90c 0500 c02f 0000 11ff 0100 0100 000b ...../.....
```

```
0x0060: 0004 0300 0102 0023 0000 1603 0303 f50b .....#. ....
```

```
0x0070: 0003 f100 03ee 0003 eb30 8203 e730 8202 .....0...0..
```

```
0x0080: cfa0 0302 0102 0209 00b7 a34d 832f be42 .....M./B
```

```
0x0090: 9f30 0d06 092a 8648 86f7 0d01 010b 0500 .0...*.H.....
```

```
0x00a0: 3081 8931 0b30 0906 0355 0406 1302 6a70 0..1.0...U...jp
```

```
0x00b0: 3110 300e 0603 5504 080c 0773 6169 7461 1.0...U...saita
```

```
0x00c0: 6d61 3111 300f 0603 5504 070c 086b 756d ma1.0...U...kum
```

```
0x00d0: 6167 6179 6131 0c30 0a06 0355 040a 0c03 agaya1.0...U...
```

```
0x00e0: 6172 7331 1430 1206 0355 040b 0c0b 696e ars1.0...U...in
```

```
0x00f0: 666f 726d 6174 696f 6e31 1430 1206 0355 formation1.0...U
```

```
0x0100: 0403 0c0b 7777 772e 6861 6374 2e6a 7031 ....www.hact.jp1
```

```
0x0110: 1b30 1906 092a 8648 86f7 0d01 0901 160c .0...*.H.....
```

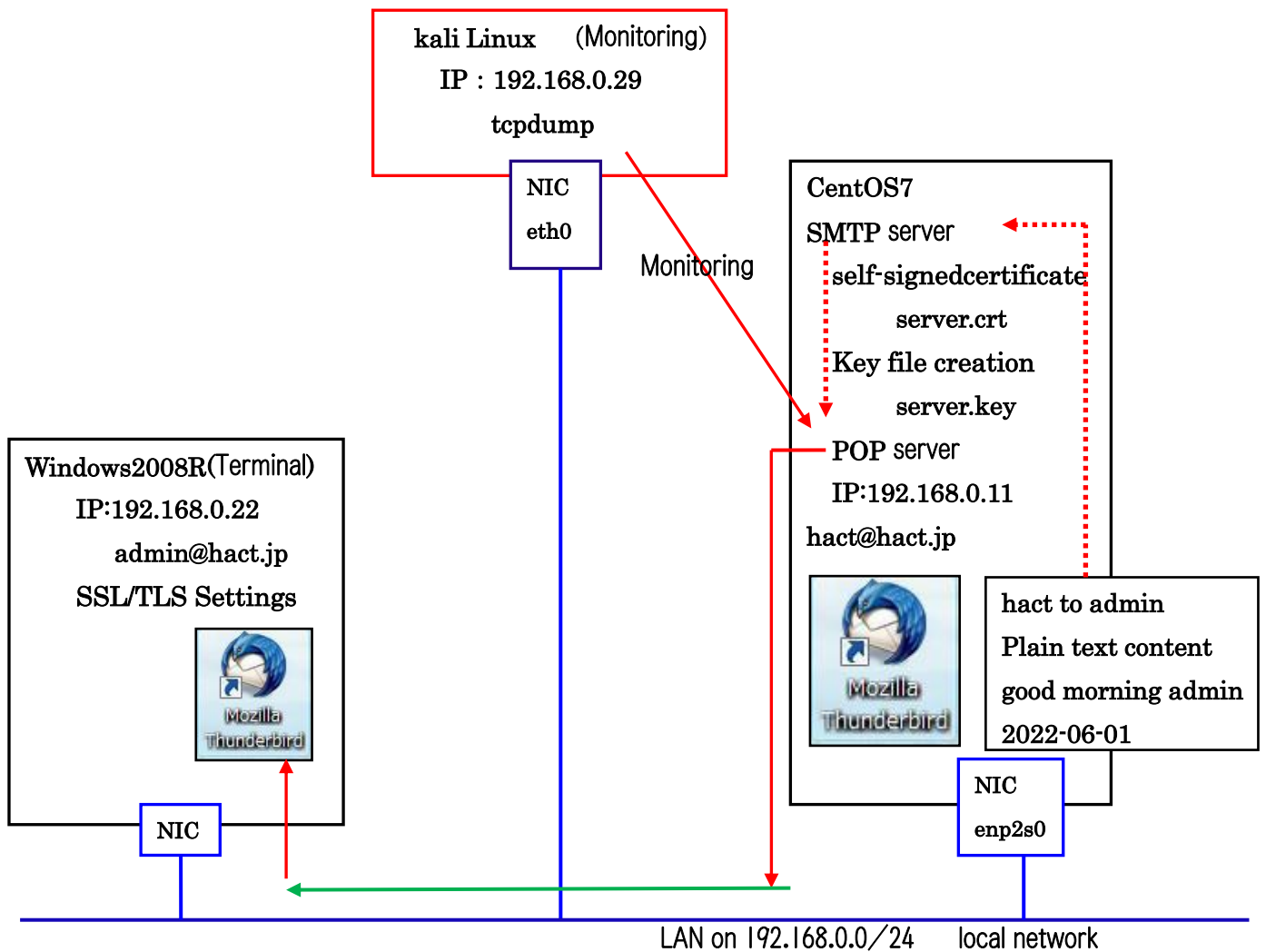
```
: Omission
```



Kitsune, if I capture an SSL/TLS encrypted email, I can see the content about the organization when the self-signed certificate (CRT file) was created, **but I can't read the body of the email or who sent it to whom. This is secure!**



The result is the same, but I'll capture the SSL/TLS encrypted POP (port number: 995) as well.



## [POP] by SSL/TLS

```
(root@kali)-[/home/kali]
└─# tcpdump -l -X port 995
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
11:41:03.311763 IP 192.168.0.22.1070 > ns.hact.jp.pop3s: Flags [S], seq 3302802315, win 8192,
options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
    0x0000:  4500 0034 02df 4000 8006 7673 c0a8 0016  E..4..@...vs....
    0x0010:  c0a8 000b 042e 03e3 c4dc c38b 0000 0000  .....
    0x0020:  8002 2000 3d25 0000 0204 05b4 0103 0308  ....=%.....
    0x0030:  0101 0402
                                     : Omission
11:41:03.311766 IP 192.168.0.22.1070 > ns.hact.jp.pop3s: Flags [P.], seq 1:518, ack 1, win 256,
length 517
                                     : Omission
    0x00a0:  0010 000e 0000 0b70 6f70 2e68 6163 742e  .....pop.hact.
    0x00b0:  6a70 0017 0000 ff01 0001 0000 0a00 0e00  jp.....
                                     : Omission
11:41:03.352632 IP ns.hact.jp.pop3s > 192.168.0.22.1070: Flags [P.], seq 1:1436, ack 518, win 237,
length 1435
                                     : Omission
    0x00b0:  1302 6a70 3110 300e 0603 5504 080c 0773  ..jp1.0...U...s
    0x00c0:  6169 7461 6d61 3111 300f 0603 5504 070c  aitama1.0...U...
    0x00d0:  086b 756d 6167 6179 6131 0c30 0a06 0355  .kumagaya1.0...U
    0x00e0:  040a 0c03 6172 7331 1430 1206 0355 040b  ....ars1.0...U..
    0x00f0:  0c0b 696e 666f 726d 6174 696f 6e31 1430  ..information1.0
                                     : Omission
```

Transfer request  
from terminal to  
POP server

Transmission  
from POP server  
to terminal  
mailer



Surely if I encrypt it with SSL/TLS, I can capture the POP server side and only the organization can see it. **It's secure!**

It's definitely better to send emails with SSL/TLS than to send them in plain text. Everyone should do this.



Yes, but... Tanuki, there is one problem: to use SSL/TLS, your ISP's (Internet Service Provider's) mail server must support SSL/TLS. In addition, the mailers of both users (e.g., Thunderbird) must be configured to use SSL/TLS in order to send and receive mail. This is a bit of a hassle to set up. So, SSL/TLS cannot be used in many cases for Web mail that is sent/received among unspecified number of people.

**So, here is my conclusion. Be careful not to send or receive passwords or other important information via e-mail!**



**Well, that's enough about how to use tcpdump.**

**In the next **episode 27**, we will cover packet analysis with Winshark.**

**Stay tuned to see how it all unfolds!**