Course on Distributed Systems at Universidad de Leon

School of Industrial, Computer and Aerospace Engineering

DSPro

Independent Study Exercises on Distributed Systems

Technical process for exercise 3 access to protocol.unileon.es

The networks that comprise the paths to host protocol.unileon.es (193.146.101.127) filter ICMP message types 13 and 14, thereby hindering any form of clock sync based that protocol. A number of strategies are available to circumvent the resulting inability to do clock sync, of which we have selected the creation of a host-to-host secure channel based on the SSH protocol. Overall, the low level detail about those tunnels is beyond the scope of our course, consequently, we'll simply deploy the tunnel and use it as what it results to be: a virtual interface to a *point-to-point network* that transparently communicates your local host and protocol.unileon.es, the target host. Thus, the ICMP packets generated by your clock sync client (By way of an IP RAW_SOCK) will travel encapsulated into SSH-encrypted messages which will not be filtered.

1. Send me an e-mail (chema.foces@unileon.es) requesting your *SSH private key*. I will send you back a web link to a file containing your *SSH private key.* That file has no extension (No characters after the dot). In this example, we assume that the file name is ***mykey12***.

   $ wget http://paloalto.unileon.es/tunnels/mykey12

2. Set the correct file permissions for the private key file (mykey12)

   # chmod **600** mykey12
   ~~# chmod **644** mykey12~~ *(Don't do this one, it's overly wrong!)*

3. Download the script that creates the tunnel and the virtual tun interface:

   $ wget http://paloalto.unileon.es/tunnels/student_tun.sh

4. Download the script that configures the newly created tun interface:

   $ wget http://paloalto.unileon.es/tunnels/student_link.sh

5. Add the *execution* attribute to both files:

   $ chmod +x student*sh

6. Do su or sudo su in your Linux and execute the script that creates the tunnel. That script needs a small integer as its first argument. You must have received that number in my response email. In this example, assume the number that I sent you back is **12**. Additionally, I will sent you back the passphrase for the private key *mykey12*:

   # ./student_tun.sh 12
   This command needs super user privileges
   First argument must be a small integer allocated specifically to you
   Keep this terminal alive while the tunnel remains established
   Enter passphrase for key 'mykey12':
   _

   If you get no error, the tunnel is set and ready. At this point, you should let the terminal keep the tunnel alive: no interaction whatsoever is necessary with it while you need the tunnel set. At this point you need a new terminal.

7. Request a new terminal and do su or sudo su and then execute the script that configures the point-to-point, SSH-based connection with 193.146.101.127:

   # ./student_link.sh 12
   This command requires super-user privileges
   First argument must be a small integer that must have been allocated to you
   4: tun12: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
   link/none
   inet 192.168.112.123 peer 192.168.112.1/32 scope global tun12
   valid_lft forever preferred_lft forever
   inet6 fe80::4205:f2b4:6284:8aeb/64 scope link stable-privacy
   valid_lft forever preferred_lft forever

   Check your point-to-point link with ping 192.168.112.1

   # ping 192.168.112.1
   PING 192.168.112.1 (192.168.112.1) 56(84) bytes of data.
   64 bytes from 192.168.112.1: icmp_seq=1 ttl=64 time=14.4 ms

64 bytes from 192.168.112.1: icmp_seq=2 ttl=64 time=13.8 ms
64 bytes from 192.168.112.1: icmp_seq=3 ttl=64 time=13.9 ms
^C
--- 192.168.112.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 13.831/14.053/14.429/0.266 ms

8. At this point, you are ready to send ICMP Timestamp requests to 193.146.101.127 and have the replies sent back to your host. *Notice that the IP address that you should send your packets to is 192.168.1**12**.1 (The underlined **12** corresponds to the number I sent you back, **12**). The local address of the point-to-point network in this case is 192.168.1**12**.123 and the address of the other end is 192.168.1**12**.1.*

9. Test that our basis icmp sync program functions ok:

    # wget paloalto.unileon.es/ds/lab/icmptimestamp.c

    # gcc -o t icmptimestamp.c

    # ./t 192.168.112.1

    Originate = 71549855, Reply Received = 71549869
    Rough Rtt = 14
    Receive = 71655885, Transmit = 71655885
    IRQ time = 0
        · backDelay = 7
        · delta = 106023

    Correction necessary upon the local clock: 106 sec, 23000 microseconds
    Not making any local clock adjustment of 106 sec, 23000 microseconds

    The ssh tunnel provides us a point-to-point connection that transports our ICMP/IP packets transparently to 193.146.101.127. The two ends of the tunnel in this technical guide have the following two IP addresses, assuming that the number that I sent you back is **12**:

    - Local address: 192.168.1**12**.123
    - Remote address: 192.168.1**12**.1

4. RMI C/S

Technical data for completing this exercise:

1. The remote java rmi object runs in 193.146.101.127 at TCP port 50002

2. The java rmiregistry where the object is stored runs in 193.146.101.127 at port 50001