NT21 Assignment 4

TCP/UDP

Task 1: TCP in Netcat

To do this assignment we will use Netcat tool which is provided in the Netkit. Netcat makes it possible to create and use TCP/UDP connections. If you want more info about Netcat you can consult Internet.

To make this assignment you should be able to communicate between two interconnected Netkit nodes. For this you can e.g. reuse the routing lab from the previous lesson and start communicating between e.g. *PC1A* and *PC2A*.

Let’s start a chat session by connecting 2 Netcat instances via a TCP connection.

To listen to the TCP connections, go to one of your simulated nodes (e.g. *PC1A*) and issue the following command:

**nc –l –p <*port\_nr*>**

This will make Netcat listen to port number that you have specified in port\_nr and accept connections.

*Note: Any port number should be ok, as long as it is not used by another application.*

To establish a TCP connection you can issue the following command from another simulated node (e.g. *PC2A*)

* **nc *<IP address of the “listening” node>* <*port\_nr of the “listening node”*>**

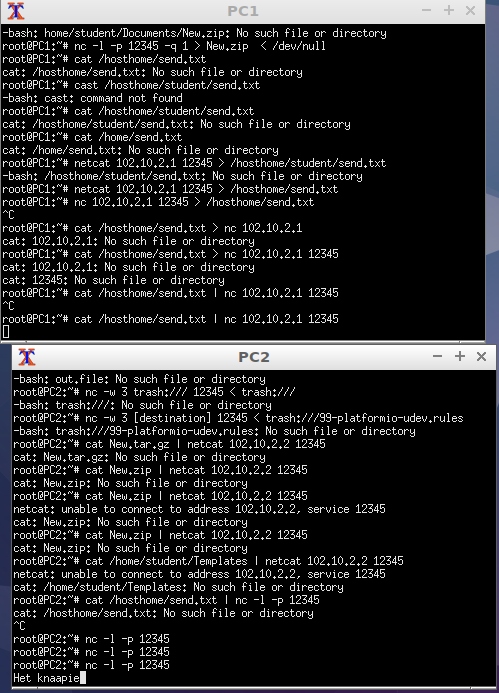
This will make a TCP connection with the listening Netcat instance.

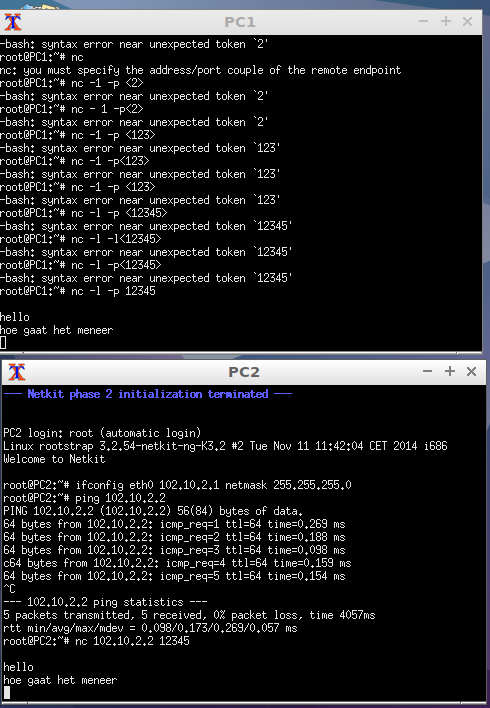
Now you can chat from one Netcat instance to the another. Try it out!

Your task:

* Netcat can also be used to copy the contents of a file from one place (file, folder, computer) to another. Find out how and try it out.

Provide screenshots of the sending and receiving command.





Task 2: TCP/UDP Experiments

In the previous task you’ve learned how to send a file with Netcat.

You’re going to use this knowledge to do a few TCP/UDP experiments.

1. Send file with TCP and UDP

Take provided file *alice.txt* and send it through Netcat. To do this, you should reuse knowledge from the previous assignment, where it was explained how to access files in Netkit and how to access files from your host environment in Netkit.

Now we’d like to see the TCP communication in Wireshark. To do that, we’ll use tcpdump command which will write all packets received on a network interface (default is network interface eth0) to a file. A file should have extension .*pcap*.

A command like this should achieve it (run it either one of the virtual linux terminals (netcat client or listener)):

***tcpdump –w <filename>.pcap -s 0 &***

*You use ‘&’ after tcpdump command to send the command execution to background. Then you can reuse command line for Netcat.   
  
To get back to your running tcpdump you can use command “fg %1”, this will bring your tcpdump program back to the foreground. Press CTRL-c to terminate the tcpdump program. On termination the tcpdump will create the requested pcap-file.*

Provide a screenshot of the Wireshark showing your captured file when you did Netcat file transfer.

How long did it take to transfer the file?

Find out how to use Netcat with UDP. Try it out. Execute the above mentioned experiment with UDP.

Provide a screenshot of the Wireshark showing your captured file when you did Netcat file transfer.

How long did it take to transfer the file?

Was the transfer successful? If not, what do you think was the reason ?

2. Send file on TCP/UDP with a smaller file

Repeat the same experiment with a smaller file *little\_alice.txt.*

Answer the following questions:

1. How long did the transfer take with TCP? Provide a screenshot from Wireshark where you can see it.
2. How long did the transfer take with UDP? Provide a screenshot from Wireshark where you can see it.
3. What can you tell about the differences?

3. Send file on TCP/UDP with a smaller file on a lossy link

Repeat subtask 2 but now on a faulty link. To simulate a faulty link you can use netem tool.

This command simulates 50% packet loss on an eth0 link:

tc qdisc add dev eth0 root netem loss 50%

To test whether your command successfully created 50% packet loss on the link, you can run a *ping* to an existing address for some time and look at the result.

After creaetion of 50% loss probability on your link repeat the experiment from subtask 2 with a smaller file *little\_alice.txt.*

Answer the following questions:

1. How long did the transfer take with TCP? Provide a screenshot from Wireshark where you can see it.
2. How long did the transfer take with UDP? Provide a screenshot from Wireshark where you can see it.
3. What can you tell about the differences?

Task 3: Flow Graph

Take any of the already created TCP wireshark traces, e.g. one used in the previous tasks. Note that a wire shark file of a non-faulty link might be easier to interpret. Go to Wireshark and select ‘*Statistics > Flow Graph*’ and then choose flowtype ‘*TCP flow*’ to draw a Sequence Diagram of the TCP message interaction that you see in Wireshark.

Provide a screenshot of this Flow Graph.

Explain what is happening duringvarious stages (begin, middle, end) of the communication. Explain SYN, SYNACK and ACK. Explain the Len, Seq and Ack numbers.

*Tip: Use Wireshark from Windows or MAC to read this file. The Linux Wireshark is less stable doing this.*

Task 4: Canvas Quiz

Execute TCP/UDP Canvas Quiz.

Task 5 (Optional): TCP SYN Flooding

Read an explanation of TCP SYN Flooding at <http://en.wikipedia.org/wiki/SYN_flood> or from some other source.

In this task you’re going to simulate this kind of DDOS attack that uses vulnerability of TCP protocol.

For this experiment you can reuse routing lab. You can use for example the *PC1A* node as the victim and *PC2A* node as an attacker.

To be able to wait for the TCP connections, use Netcat command to wait for the TCP connections on a specific port at the victim node.

To simulate TCP SYN flood traffic from the attacker node, you can use the “hping3” tool which is part of your Netkit nodes.

Before you start the attacker command, don’t forget to sniff the traffic with tcpdump command and write the output to a pcap file (as you’ve done previously in this assignment).

A command to be issued at the attacker node can look like:

hping3 --rand-source <IP\_ADDRESS\_OF\_VICTIM> --flood –S –L 0 –p <PORT\_NR\_OF\_VICTIM>.

Wait about 10 seconds, stop hping3 and tracing.

Now you should be able to analyze the trace. You should be able to see spoofed source IP address.

Analyze your trace. Find out how many SYNs, SYN+ACKs and ACKs you can see. Explain what do these numbers tell you about SYN attack.

Consult Internet to find out how another transport protocol – SCTP - solves TCP SYN flooding problem. Give a short explanation of how it is implemented in SCTP.