Universiteit Antwerpen

INHOUD

Title: UDP PACKETS TO NODEMCU

Mohammad, Amir

Intro

How can we make a connection between 2 computers? What kind of connections are they and what are their benefits? But more importantly in this case, how can we make a connection between our Ubuntu server and the nodeMCU?
In this report we will find an answer for these questions.

Materials and Methods

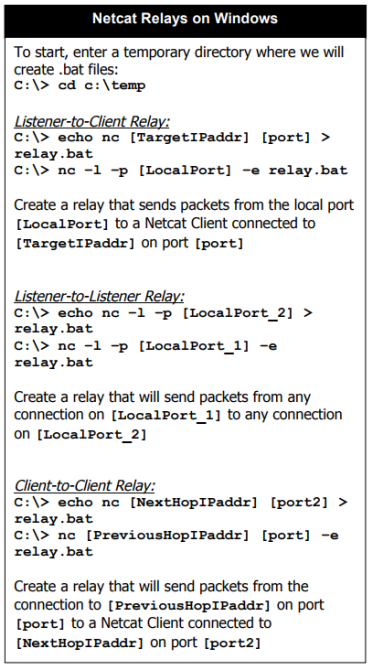
Results

Connections between 2 computers can be done wireless (Antenna with receiver and transmitter) or wired with a cable (UTP most often used). The ip address of both pc's should be in the same subnet to have a local connection.
Making connections can be done in different ways. CO (Connection Oriented) is a good way to make sure the package will be delivered. However, this takes more time to make the connection and to end the connection. The connection has to be set up by sending starting frames and displaying eachothers buffers to check either you can send your packages or receive more packages from the other side. In case of master-slave connection ofcourse you will only receive or send packages depending on your rights to do so. TCP is a typical example of CO connection in Layer 4. In CL (Connection Less) you don't have these handshaking frames, no buffers, just straight it sends its frame and hopefully it will reach its destination. The advantage of this kind of connection is that you will send or receive the frames very quickly but you are not guaranteed that your package will reach its destination, there might occur some faults. This type of connection is highly recommended in connections that have small chance of faults occurring or data that is permissible to have some errors. We are choosing the UDP CL connection, because of its fast speed and we are most likely to have no faults in the connection.
Commonly when using any linux OS there is use of the bash shell. Bash shell is a command processor using a text window to execute commands typed by the user. Sending UDP packets from the ubuntu server should be possible by the following command:
Echo "My data" >/dev/[ip-adres]/[port]

Another way of sending udp packages via ubuntu can be done by using netcat.

```
$ nc [options] [TargetIPaddr] [port(s)]
```

For instance, you can use it like this: nc -l -p [LocalPort] -e [FileWithCommandsToBeExecuted]
Check the datasheet below for more information on how to use this command



Netcat Relays on Windows

To start, enter a temporary directory where we will create .bat files:
`C:\> cd c:\temp`

Listener-to-Client Relay:
```
C:\> echo nc [TargetIPaddr] [port] >
relay.bat
C:\> nc -l -p [LocalPort] -e relay.bat
```

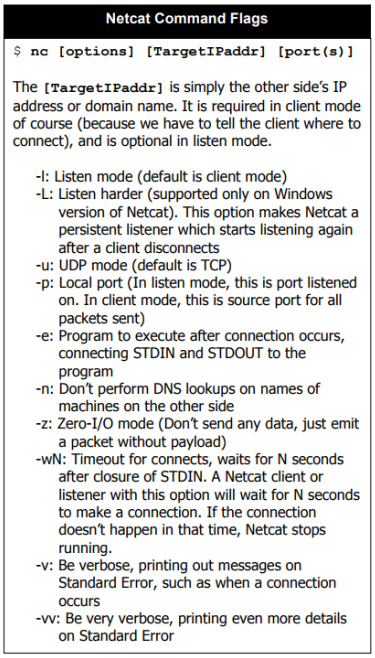
Create a relay that sends packets from the local port `[LocalPort]` to a Netcat Client connected to `[TargetIPaddr]` on port `[port]`

Listener-to-Listener Relay:
```
C:\> echo nc -l -p [LocalPort_2] >
relay.bat
C:\> nc -l -p [LocalPort_1] -e
relay.bat
```

Create a relay that will send packets from any connection on `[LocalPort_1]` to any connection on `[LocalPort_2]`

Client-to-Client Relay:
```
C:\> echo nc [NextHopIPaddr] [port2] >
relay.bat
C:\> nc [PreviousHopIPaddr] [port] -e
relay.bat
```

Create a relay that will send packets from the connection to `[PreviousHopIPaddr]` on port `[port]` to a Netcat Client connected to `[NextHopIPaddr]` on port `[port2]`



Netcat Command Flags

`$ nc [options] [TargetIPaddr] [port(s)]`

The `[TargetIPaddr]` is simply the other side's IP address or domain name. It is required in client mode of course (because we have to tell the client where to connect), and is optional in listen mode.

-l: Listen mode (default is client mode)
-L: Listen harder (supported only on Windows version of Netcat). This option makes Netcat a persistent listener which starts listening again after a client disconnects
-u: UDP mode (default is TCP)
-p: Local port (In listen mode, this is port listened on. In client mode, this is source port for all packets sent)
-e: Program to execute after connection occurs, connecting STDIN and STDOUT to the program
-n: Don't perform DNS lookups on names of machines on the other side
-z: Zero-I/O mode (Don't send any data, just emit a packet without payload)
-wN: Timeout for connects, waits for N seconds after closure of STDIN. A Netcat client or listener with this option will wait for N seconds to make a connection. If the connection doesn't happen in that time, Netcat stops running.
-v: Be verbose, printing out messages on Standard Error, such as when a connection occurs
-vv: Be very verbose, printing even more details on Standard Error

SANS INSTITUTE

Netcat Cheat Sheet
By Ed Skoudis
POCKET REFERENCE GUIDE
http://www.sans.org

Purpose

This cheat sheet provides various tips for using Netcat on both Linux and Unix, specifically tailored to the SANS 504, 517, and 560 courses. All syntax is designed for the original Netcat versions, released by Hobbit and Weld Pond. The syntax here can be adapted for other Netcats, including ncat, gnu Netcat, and others.

Fundamentals

Fundamental Netcat Client:
`$ nc [TargetIPaddr] [port]`

Connect to an arbitrary port `[port]` at IP Address `[TargetIPaddr]`

Fundamental Netcat Listener:
`$ nc -l -p [LocalPort]`

Create a Netcat listener on arbitrary local port `[LocalPort]`

Both the client and listener take input from STDIN and send data received from the network to STDOUT

Link datasheet Netcat: https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
We can also use the following application to send packages, the package sender:
https://github.com/esp8266/Arduino/blob/master/doc/esp8266wifi/udp-examples.rst

Information & conclusion

See Results

Referencelist

Link datasheet Netcat: https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
https://github.com/esp8266/Arduino/blob/master/doc/esp8266wifi/udp-examples.rst