

The today's laboratory task is to implement a python script called **cryptocat** which simulates the linux command **netcat** or **nc** (which can read/write data using tcp/ip connections, see below) extended with the functionalities necessary to encrypt/decrypt transmitted data. This can be done using the **openssl enc** command.

With the command **netcat** you can connect to a remote host (or locally on localhost) and connect to *stdin* and *stdout* stream data sent or received from remote. In particular, the command **netcat** or **nc** can be used:

- in server mode with the syntax:

```
netcat -l port or netcat -l -p port
```

-l is the option to activate *listen* mode that puts server in "listening" state on a given TCP port.

port is an integer that specifies the *port number* on which the server is listening

- in client mode with the syntax:

```
netcat <hostname> port
```

hostname is the *IP address* of the server to connect to;

port is an integer that specifies the *port number* to be connected to.

In "listen" mode **netcat** listens on a port until some remote client connects to it (clients may also be local). At this point it will be established a link between the client and server.

In client mode, instead, **netcat** connects to a remote server and can send data.

The easier and primary netcat use is that of a very simple **server-client** data transfer.

For example, one can create on pc1 (which has IP address 192.168.0.1) a server listening on port 3333 by typing the command:

```
netcat -l 3333
```

On pc2 instead one executes the command:

```
netcat 192.168.0.1 3333
```

with which you connect by using **netcat** to pc1 on port 3333.

At this point everything you write in pc1's console will be transferred to pc2 and vice versa, because the standard input of a pc is connected to the standard output of the other. Transferring a file is as easy as

```
netcat -l 3333 > destinationfile
```

```
netcat 192.168.0.1 3333 < sourcefile
```

Respectively on the server and client machines.

Using the command **openssl enc** one can encrypt and decrypt data blocks with a large set of cryptographic algorithms. Specifically, the command **openssl enc** can be used with the syntax:

```
openssl enc [-algorithm] [-e] [-d] [-k key] [-in file] [-out file]
```

- algorithm is used to indicate the algorithm used to perform the symmetric encryption operation (the full list of available algorithms displays with **openssl enc -h**)

-e operation to be done is *encryption* (excludes the **-d** option)

-d operation to be done is *decryption* (excludes the **-e** option)

-k key for symmetric encryption operation

-in file the input file to encrypt (takes *stdin* if not specified)

-out file the output file encrypted (takes *stdout* if not specified)

-base64 to have the output in base64 (useful if preparing data to be sent as text)

The **cryptocat** script should be invoked as follows:

```
cryptocat [options][hostname] port
```

- `options` is a parameter that includes a list of options as:

- o `-l`, if present, will activate listen mode, that is, server mode
- o `-k key` for setting a key for encryption operation
- o `-a algorithm`, if present, can be used to indicate the algorithm used to perform the encryption/decryption operation otherwise we assume a default algorithm;

- `hostname` an optional parameter that indicates - when cryptocat is invoked in client mode - the IP address of the server to connect to

- `port` is an integer that specifies the *port number* to which to connect as client or listen as

server If executed in client mode (without the option `-l`), the script:

1. reads from *stdin* and encrypts data stream that has been read using the command `openssl enc` (which will be invoked with parameters specified, for example, `key, algorithm`)
2. sends the encrypted data stream (in base64) to server `hostname` on port `port`

Using the `netcat` command

If executed in server mode (with the option `-l`), the script:

1. invokes the `netcat` command in *listen* mode, then listens on port `port` until it doesn't receive data stream from a client.
2. reads data stream from client and writes it decrypted on *stdout* using `openssl enc` command (which will be invoked with parameters specified, for example, `key, algorithm`).

Tip : recall that it is possible to pipe the linux commands `cat`, `openssl enc` and `netcat` for (1) providing a data stream, (2) encrypting, and (3) sending to the remote server; on the server side it is possible to pipe the commands `netcat` and `openssl enc` for (1) listening for a data stream from client, and (2) decrypting it.

Run the script **cryptocat** in both modes (two different shell) analyzing the data traffic between server and client with the **Wireshark** software.