Exercises in Secure Mobile Systems

FH Hagenberg, WS 2022
FH-Prof. DI Dr. Erik Sonnleitner, OSCP

Exercise 4: Network scanning & Encryption programming

Exercise 4: Network scanning & Encryption programming

(1) Use any netcat flavor to perform the following tasks:
- (1.1) Create a bind-shell on the host netcat is in server (listening) mode. A bind-shell allows netcat clients to use a server-side shell (execute commands on the server).
- (1.2) Create a reverse-shell. A reverse shell is the counterpart of a bind-shell and allows the server to back-connect to the client
- (1.3) Can these shells be used when server and client are using different OSs?
- (1.4) Can (1.1) and/or (1.2) used with UDP?

(2) Construct server and client netcat command-lines which resemble simple a text-chat.
- Chat must be accessible to multiple clients simultaneously
- All clients must receive all messages from all other clients
- All traffic must be encrypted at all times
- Chat server must only allow predefined IPs
- Server must keep a central chat log

(3) What does the following command-line do? **`while $(ncat -lp 8080 -c 'ncat localhost 80'); do true; done`**

(4) Use **`ncat`** to fetch your e-mails from your FH mail account via IMAP. See **`https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol`** for how the IMAP protocol works.

Exercise 4: Network scanning & Encryption programming

(5) Use **`nmap`** to answer the following questions regarding the host **`delta-xi.net`**:
- Which ports are used for SSH?
- What's the RSA public key for SSH?
- When does the IMAP certificate expire?
- For which domain, other than **`delta-xi.net`**, does the host accept e-mails for?
- What's the welcome message the SMTP server provides?

(6) Use **libsodium** to develop a simple message en-/decryption command-line client called **`msgcrypt`**. All messages should be encrypted using AES-256 with CBC block mode. Input data is read from **`stdin`** (i.e. via a pipe), and processed depending on arguments, e.g.:
- **`$ echo "hello world" | ./msgcrypt -key key.txt -enc`**
  `aGVsbG8gd29ybGQK`
- **`$ echo "aGVsbG8gd29ybGQK" | ./msgcrypt -key key.txt -dec`**
  `hello world`

- encrypted data must be represented in base64 format in order not to print binary data on the command-line!
- Use Netcat in client-mode to send an encrypted message to one of your colleagues; he/she uses netcat in server mode and use his/her libsodium implementation to decrypt the message. If you coded carefully, this should work without problems - supply a screenshot.