Sharif University of Technology
Department of Computer Engineering

Advance Network Security
Instructor: Dr.MohammadMehdi SadeghZadeh *2 Aban, 1402*

Homework 1[*]

Revisiting previous course material helps solidify fundamental concepts. The goal of this assignment is to explore the benefits of reviewing ”Network Security” course materials to promote deeper understanding , and ultimately contributes to overall academic achievement.

1 Part A: Fundamentals on Cryptography

1.1 DES vs. AES

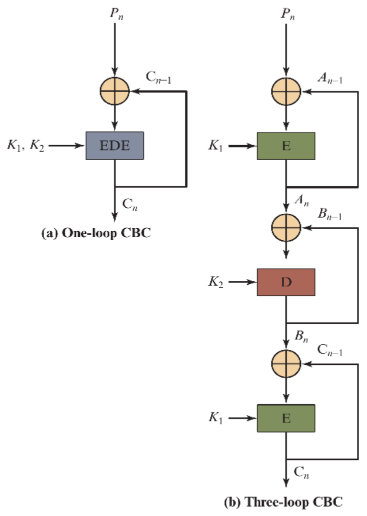
(a) For each of the following elements of DES, indicate the differences with the compa- rable element in AES: Key size, Block size, S-box, Key expansion function , and Initial and final permutation

(b) Consider an available implmenentation of DES and AES. Write your own program to check the Avalanche effect in the above DES algorithm. To do so, you should change every single bit of input and then check its effect on output. Check the Avalanche effect in AES and compare it with DES.

1.2 Cipher Block Chaining (CBC)

Two proposals are showin Figure 1 in order to build a hardware device which do block encryption in the CBC mode using TripleDES algorithm.

Figure 1: Use 3DES in CBC Mode



(a) Explain in each of the following situation , which of the two would you choose and why. 1) security is matter 2) performance is matter

*Acknowledgement: This homework was orginally developed by A.Ameli, F.babaTabar, P.Chavoshian, R.Eskandari, M.H.Jangholi, and E.ZarinKia ,

(b) Can you suggest any security improvement to either option in Figure 1 , using only three DES chips and some number of XOR functions? Assume you are still limited to two keys.

1.3 Secure Hash

The main motivation for this problem is to construct a hash function based on using a CBC technique but without using the secret key for the sake of minimization of design and implementation effort.

(a) Consider the following scheme as example. A message M is devided into N fixed-size blocks ($M_1, M_2, ..., M_N$). Then a symmetric encryption system such as DES will be used to compute the hash code H as:

$H_0 = initialvalue$

$H_i = H_{i-1} \oplus E(M_i, H_{i-1})$

$H = H_N$

Assume that DES is used as the encryption algorithm. Could you provide an attack against this hashing scheme?

(b) Consider the following variation of the scheme above with the following formula: $H_i = M_i \oplus E(H_{i-1}, M_i)$ Show that a similar attack succeed against this scheme.

Hint: According to the complementarity property of DES: If Y = E(K, X), then $\acute{Y} = E(\acute{K}, \acute{X})$.

1.4 Security Protocols

The main motivation for this problem is to construct a hash function based on using a CBC technique but without using the secret key for the sake of minimization of design and implementation effort.

(a) Consider the following scheme as example. A message M is devided into N fixed-size blocks ($M_1, M_2, ..., M_N$). Then a symmetric encryption system such as DES will be used to compute the hash code H as:

$H_0 = initialvalue$

$H_i = H_{i-1} \oplus E(M_i, H_{i-1})$

$H = H_N$

Assume that DES is used as the encryption algorithm. Could you provide an attack against this hashing scheme?

(b) Consider the following variation of the scheme above with the following formula: $H_i = M_i \oplus E(H_{i-1}, M_i)$ Show that a similar attack succeed against this scheme.

Hint: According to the complementarity property of DES: If Y = E(K, X), then $\acute{Y} = E(\acute{K}, \acute{X})$.

1.5 IPSec

Suppose Ali sends packets to Hossein using TCP over IPSec. suppose that the TCP acknowledgement from Hossein is lost. So Ali's TCP sender will assume the corresponding data packet was lost, and thus retransmits the packet. Does the IPSec at Hossein side considers the retransmitted TCP packet be as a replay packet and discard it? Please briefly explain your answer.

2 Part B: TCP Vulnerabilities And Exploits

2.1 UBUNTU SETUP

First download seed-Ubuntu and VirtualBox 6.1.16. this Manual guides you to install them.
NOTE: All usernames are "seed" and passwords are "dees".

2.2 DOCKER SETUP

First download lab.zip. Unzip the given lab file. run the terminal in the lab folder. build docker file with `$ dcbuild` command. run docker file with `$ dcup` command (This command downloads roughly 100MB of packages for the first time you run it). Open a new tab in the terminal and use `$ dockps` to see the ID of containers. open three new tabs in the terminal and use `$ docksh ”ID”` to connect to each container(user, victim, attacker).
NOTE: Please shutdown docker with `$ dcdown` every time you finish.

2.3 Task 1: TCP Session Hijacking

In this section, you have to run a session hijacking attack on two victims (user and victim). you have to connect from user to victim by `$ telnet ”victimID”`. you can use `$ netstat -nat` on the victim to see if the user is connected. open wireshark and search for `”IP victim” and tcp and port ”Port victim”`. then use the `$ ls` command on the user terminal. Now you have to use `$ cat > secret` and write your full name inside it. you can check existing files using the `$ ls` command (you can also delete these files using the `$ rm -f ”file_name”` command). use this command `$ cat ”file_name”` to see content inside the secret file. Now check the last TCP connection in Wireshark. you need a sequence number and an acknowledgment number to run this attack.
Your Task: you have to create a Python file named session_hijack.py inside the volumes folder (lab/volumes/session_hijack.py). you can create the file code by opening the terminal inside the volumes folder and running the `$ touch session_hijack.py` command. you can also open and edit this code using the `$ gedit session_hijack.py` command. in this code, you can only use **scapy.all** library. your task is to write a code that runs on the attacker terminal and the goal is to read the secret file that you created earlier. after you write the code use the `$ cd volumes` command on the attacker terminal. first, run the `$ nc -l 9090 &` command in the attacker container then use `$ python3 session_hijack.py` command to run the code.
NOTE: for session_hijack.py you have to manually write the needed parameter. don't use commands to automatically fill sequence and acknowledgment numbers.

2.4 Task 2: Creating Reverse Shell using TCP Session Hijacking

In the previous section, you get familiar with session hijacking for reading files in the victim's container. You can set up a reverse shell if you have access to the target machine, which is the telnet server in our setup, but in this task, you do not have such access. Your task is to launch a TCP session hijacking attack on an existing telnet session between a user and the target server. You need to inject your malicious command into the hijacked session, so you can get a reverse shell on the target server.
You have to write a Python code named reverse_shell.py in volumes folder only using **scapy.all** library. **Important**: unlike the previous section, in this section, you have to use commands that automatically fill parameters for you (like IPs and Ports and seq and ack number). Now make sure the user is connected to the victim via telnet. After writing the code, you have to go to the attacker terminal and run the `$ nc -l 9090 &` command, and use `$ python3 reverse_shell.py &` to run the attack. now go to the user terminal and run the `$ ls` command a couple of times until the terminal freezes. Now go back to the attacker terminal. you have to see that the attacker is connected to the victim. Press enter and run `$ jobs`. you can see the nc -l 9090 index. use `$ fg “index` to connect to the victim. now you have access to write and delete files of the victim's machine inside the attacker terminal. write a new file named new_secret in the victim's machine from the attacker terminal using `$ cat > new_secret` and write you student ID inside it. Now go back to the user terminal and run `$ ls` command to see if the new file added to the files and read it using `$ cat new_secret`. (if the user terminal is still frozen, go to the victim terminal and terminate the connection, and connect again.)

2.5 HW Submission

Deliverables:

You should submit .pdf file containing answers to theorical questions (Q1.1..1.5), along with the required program source code for Q1.b , its detail report, as well as the explanation of how to run your program.

You must submit a detailed lab report for Q2, with screenshots for Terminals and Wireshark, to describe what you have done and what you have observed, as well as the program source code. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.

Finally, submit all of your answers in .zip file on the Quera course page with the following format:

HW[HWNo]-[FamilyName]-[stdNo] .zip (For example, HW3-Hoseini-401234567.zip)