

Assignment 3

Eavesdrop attack

Roll no.C16

An eavesdrop attack (passive sniffing) is when an attacker listens to network traffic between two hosts to intercept sensitive data (messages, credentials, etc.) without altering the traffic. Unencrypted protocols (plain UDP/TCP) are vulnerable because packet payloads can be read by anyone capturing the link.

Tools & environment used

- **OS:** Linux(VirtualBox)
- **CPP** (for UDP client/server)
- tcpdump

Code:#

```
mkdir -p ~/udp-  
eavesdrop && cd  
~/udp-eavesdrop
```

```
# Server
```

```
cat > UDPServer.cpp  
<<'EOF'  
#include <iostream>  
#include <cstring>  
#include <arpa/inet.h>  
#include <unistd.h>
```

```
#define  
BUFFER_SIZE 1024
```

```

std::string
caesar_decrypt(const
std::string &s, int shift)
{
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i <
out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c
<= 'Z') {
            out[i] = char((c
- 'A' - shift + 26) % 26
+ 'A');
        } else if (c >= 'a'
&& c <= 'z') {
            out[i] = char((c
- 'a' - shift + 26) % 26
+ 'a');
        } else {
            // leave other
characters as-is
(spaces, punctuation,
digits)
            out[i] = c;
        }
    }
    return out;
}

```

```

std::string
caesar_encrypt(const
std::string &s, int shift)
{
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i <
out.size(); ++i) {
        char c = out[i];

```

```

        if (c >= 'A' && c
<= 'Z') {
            out[i] = char((c
- 'A' + shift + 26) % 26
+ 'A');
        } else if (c >= 'a'
&& c <= 'z') {
            out[i] = char((c
- 'a' + shift + 26) % 26
+ 'a');
        } else {
            out[i] = c;
        }
    }
    return out;
}

```

```

int main(int argc,
char* argv[]) {
    int port = 9876; //
default port
    int shift = 3; //
default Caesar shift
    if (argc > 1) port =
atoi(argv[1]);
    if (argc > 2) shift =
atoi(argv[2]);

    int sockfd;
    struct sockaddr_in
serverAddr,
clientAddr;
    char
buffer[BUFFER_SIZE]
;
    socklen_t addrLen
= sizeof(clientAddr);

    if ((sockfd =
socket(AF_INET,

```

```

SOCK_DGRAM, 0)) <
0) {
    perror("Socket
creation failed");

    exit(EXIT_FAILURE);
}

```

```

serverAddr.sin_family
= AF_INET;

```

```

serverAddr.sin_addr.s
_addr =
INADDR_ANY;
    serverAddr.sin_port
= htons(port);

```

```

    if (bind(sockfd,
(const struct
sockaddr*)&serverAd
dr, sizeof(serverAddr))
< 0) {
        perror("Bind
failed");
        close(sockfd);

```

```

    exit(EXIT_FAILURE);
}

```

```

    std::cout << "UDP
server running on port
" << port << " with
Caesar shift " << shift
    << " (type
'exit' from client to
stop)\n";

```

```

    while (true) {
        memset(buffer,

```

```

0, BUFFER_SIZE);
    int n =
recvfrom(sockfd,
buffer,
BUFFER_SIZE, 0,
(struct
sockaddr*)&clientAddr
, &addrLen);
    if (n < 0) {

perror("recvfrom
failed");
    continue;
    }

    std::string
encrypted_msg(buffer
, n);
    std::string
decrypted_msg =
caesar_decrypt(encry
pted_msg, shift);

    std::cout <<
"Received
(ciphertext): " <<
encrypted_msg << "
from "
        <<
inet_ntoa(clientAddr.si
n_addr) << ":" <<
ntohs(clientAddr.sin_p
ort) << "\n";
    std::cout <<
"Decrypted (plaintext):
" << decrypted_msg
<< "\n";

    if
(decrypted_msg ==

```

```

"exit") {
    std::string
bye_plain = "Server
shutting down as
requested.";
    std::string
bye_encrypted =
caesar_encrypt(bye_p
lain, shift);
    sendto(sockfd,
bye_encrypted.c_str(),
bye_encrypted.size(),
0, (struct
sockaddr*)&clientAddr
, addrLen);
    std::cout <<
"Exit command
received. Shutting
down server.\n";
    break;
}

```

```

    std::string
response_plain =
"Server ACK: " +
decrypted_msg;
    std::string
response_encrypted =
caesar_encrypt(respo
nse_plain, shift);
    sendto(sockfd,
response_encrypted.c
_str(),
response_encrypted.s
ize(), 0, (struct
sockaddr*)&clientAddr
, addrLen);
}

```

```

close(sockfd);

```

```
    return 0;
}
EOF
```

```
# Client UDP Client
```

```
cat > UDPClient.cpp
<<'EOF'
#include <iostream>
#include <cstring>
#include <arpa/inet.h>
#include <unistd.h>
```

```
#define
BUFFER_SIZE 1024
```

```
std::string
caesar_encrypt(const
std::string &s, int shift)
{
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i <
out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c
<= 'Z') {
            out[i] = char((c
- 'A' + shift + 26) % 26
+ 'A');
        } else if (c >= 'a'
&& c <= 'z') {
            out[i] = char((c
- 'a' + shift + 26) % 26
+ 'a');
        } else {
            out[i] = c;
        }
    }
    return out;
```

```
}
```

```
std::string  
caesar_decrypt(const  
std::string &s, int shift)  
{  
    std::string out = s;  
    shift = shift % 26;  
    for (size_t i = 0; i <  
out.size(); ++i) {  
        char c = out[i];  
        if (c >= 'A' && c  
<= 'Z') {  
            out[i] = char((c  
- 'A' - shift + 26) % 26  
+ 'A');  
        } else if (c >= 'a'  
&& c <= 'z') {  
            out[i] = char((c  
- 'a' - shift + 26) % 26  
+ 'a');  
        } else {  
            out[i] = c;  
        }  
    }  
    return out;  
}
```

```
int main(int argc,  
char* argv[]) {  
    const char*  
serverIP =  
"127.0.0.1";  
    int serverPort =  
9876;  
    int shift = 3; //  
default Caesar shift
```

```
    if (argc >= 2)  
serverIP = argv[1];
```



```
    if (argc >= 3)
serverPort =
atoi(argv[2]);
    if (argc >= 4) shift =
atoi(argv[3]);
```

```
    int sockfd;
    struct sockaddr_in
serverAddr;
    char
buffer[BUFFER_SIZE]
;
```

```
    if ((sockfd =
socket(AF_INET,
SOCK_DGRAM, 0)) <
0) {
        perror("Socket
creation failed");

exit(EXIT_FAILURE);
    }
```

```
serverAddr.sin_family
= AF_INET;
    serverAddr.sin_port
= htons(serverPort);
    inet_pton(AF_INET,
serverIP,
&serverAddr.sin_addr
);
```

```
    socklen_t addrLen
= sizeof(serverAddr);
```

```
    std::cout << "UDP
client started. Sending
to " << serverIP << ":"
<< serverPort
```

```
        << " with  
Caesar shift " << shift  
<< "\n";
```

```
    std::cout << "Type  
messages and press  
Enter. Type 'exit' to  
stop server and  
exit.\n";
```

```
    while (true) {  
        std::cout <<  
"You: ";  
        std::string plain;
```

```
std::getline(std::cin,  
plain);
```

```
        std::string  
encrypted =  
caesar_encrypt(plain,  
shift);  
        sendto(sockfd,  
encrypted.c_str(),  
encrypted.size(), 0,  
(struct  
sockaddr*)&serverAd  
dr, addrLen);
```

```
        int n =  
recvfrom(sockfd,  
buffer,  
BUFFER_SIZE, 0,  
(struct  
sockaddr*)&serverAd  
dr, &addrLen);
```

```
        if (n < 0) {  
  
perror("recvfrom  
error");  
        break;
```

```

    }
    std::string
resp_encrypted(buffer
, n);
    std::string
resp_plain =
caesar_decrypt(resp_
encrypted, shift);

    std::cout <<
"Server (ciphertext): "
<< resp_encrypted <<
""\n";
    std::cout <<
"Server (decrypted): "
<< resp_plain << ""\n";

    if (plain == "exit")
break;
}

close(sockfd);
return 0;
}
EOF

```

Put `UDPServer.cpp` and `UDPClient.cpp` in the same folder, e.g.
`mkdir -p ~/udp-eavesdrop && cd ~/udp-eavesdrop`

Open **Command Prompt** (normal) and navigate to the folder

Compile

```
g++ UDPServer.cpp -o UDPServer
```

```
g++ UDPClient.cpp -o UDPClient
```

Run 3 terminals

Terminal 1 — server (runs on default port 9876 and shift 3)
cd ~/udp-eavesdrop
./UDPServer 9876 3

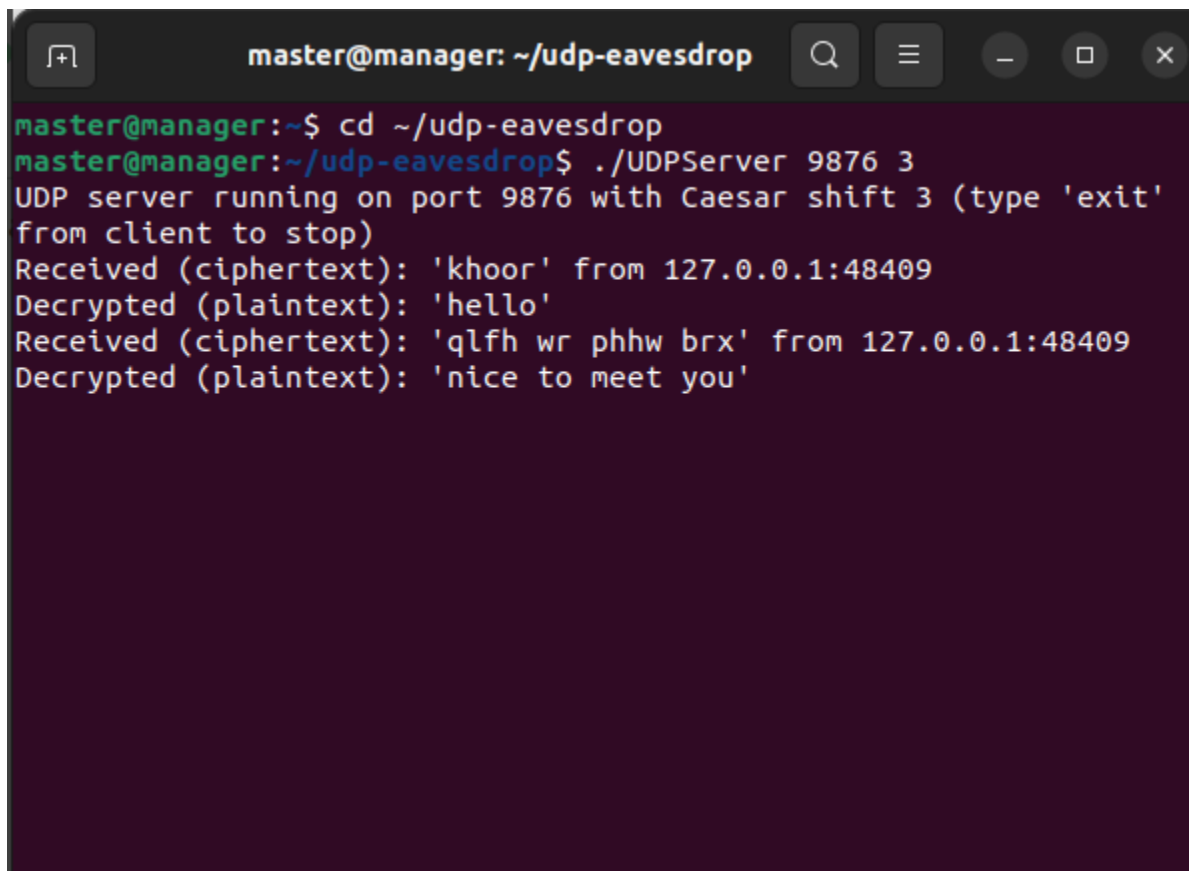
Terminal 2 — client:
cd ~/udp-eavesdrop
./UDPClient 127.0.0.1 9876 3

Type messages in the client. The client will show:

- the ciphertext sent (not shown to server; only sent),
- the server's ciphertext reply,
- and the decrypted server reply.
-

Terminal 3 — attacker (tcpdump)
sudo tcpdump -i lo -nn -s 0 udp port 9876 -A

1.server terminal output

A screenshot of a terminal window titled 'master@manager: ~/udp-eavesdrop'. The terminal shows the following output:

```
master@manager:~$ cd ~/udp-eavesdrop
master@manager:~/udp-eavesdrop$ ./UDPServer 9876 3
UDP server running on port 9876 with Caesar shift 3 (type 'exit'
from client to stop)
Received (ciphertext): 'khoor' from 127.0.0.1:48409
Decrypted (plaintext): 'hello'
Received (ciphertext): 'qlfh wr phhw brx' from 127.0.0.1:48409
Decrypted (plaintext): 'nice to meet you'
```

2. Client terminal output

```
master@manager: ~/udp-eavesdrop
master@manager:~$ cd ~/udp-eavesdrop
master@manager:~/udp-eavesdrop$ ./UDPClient 127.0.0.1 9876 3
UDP client started. Sending to 127.0.0.1:9876 with Caesar shift 3
Type messages and press Enter. Type 'exit' to stop server and exit.
You: hello
Server (ciphertext): 'Vhuyhu DFN: khood'
Server (decrypted): 'Server ACK: hello'
You: nice to meet you
Server (ciphertext): 'Vhuyhu DFN: qlfh wr phhw brx'
Server (decrypted): 'Server ACK: nice to meet you'
You: █
```

3. Attacker terminal output

```
master@manager: ~
master@manager:~$ sudo tcpdump -i lo -nn -s 0 udp port 9876 -A
[sudo] password for master:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:04:07.252419 IP 127.0.0.1.48409 > 127.0.0.1.9876: UDP, length 5
E..!..@.@..9.....&.... khood
14:04:07.254174 IP 127.0.0.1.9876 > 127.0.0.1.48409: UDP, length 17
E..-..@.@.,.....&.....,Vhuyhu DFN: khood
14:08:59.060090 IP 127.0.0.1.48409 > 127.0.0.1.9876: UDP, length 16
E...7@.@.9.....&....+qlfh wr phhw brx
14:08:59.061950 IP 127.0.0.1.9876 > 127.0.0.1.48409: UDP, length 28
E..8.8@.@.9{.....&....$.7Vhuyhu DFN: qlfh wr phhw brx
```